

TEMA 3.

PUNTERS.

TEORIA.

FONAMENTS DE PROGRAMACIÓ II

CURS: 2024-25

GRAUS: GEI, GEI-Biotec

UNIVERSITAT ROVIRA I VIRGILI

PUNTERS

Punters

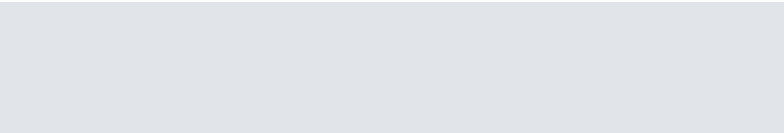
Funcionament

algorisme
var

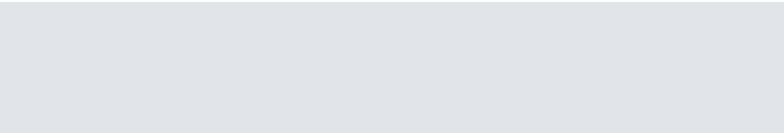
fvar
inici

falgorisme

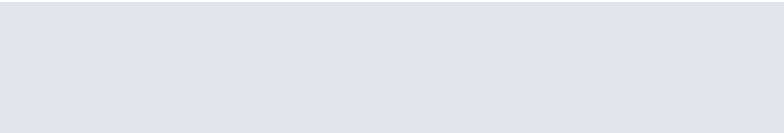
@ 1000



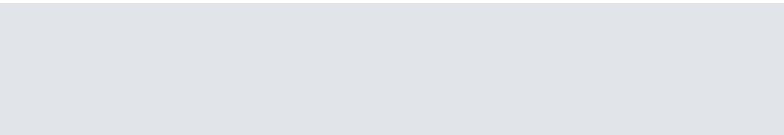
@ 1001



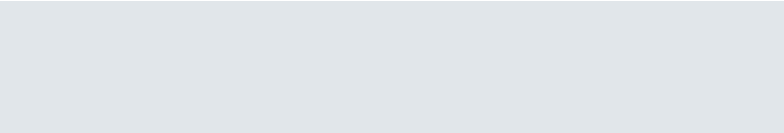
@ 1002



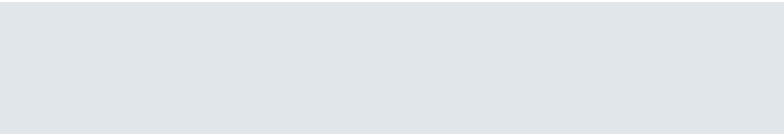
@ 1003



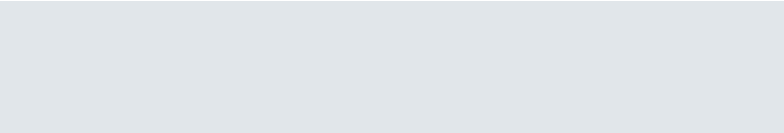
@ 1004



@ 1005



@ 1006



Punters

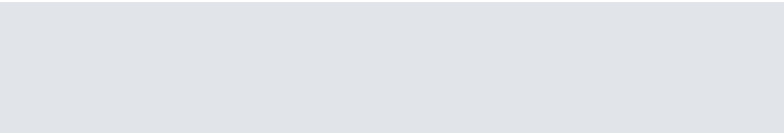
Funcionament

```
algorisme exemple és
var
    v : enter;

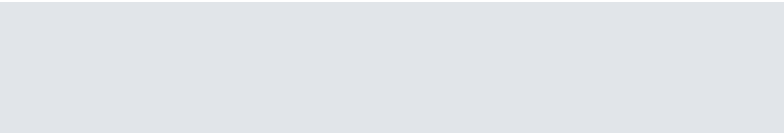
fvar
inici

falgorisme
```

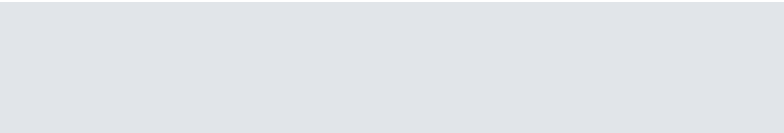
@ 1000



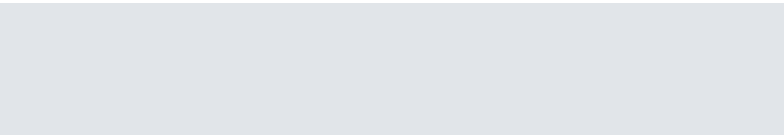
@ 1001



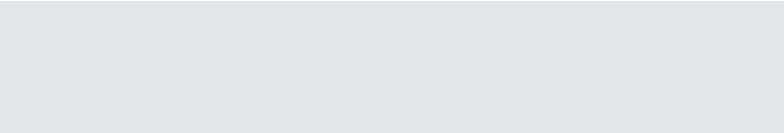
@ 1002



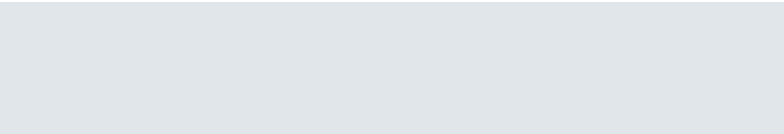
@ 1003



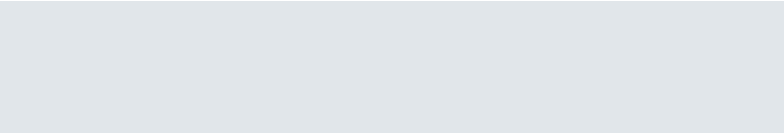
@ 1004



@ 1005



@ 1006



Punters

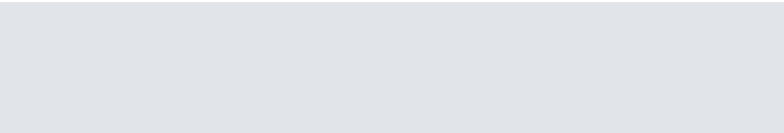
Funcionament

```
algorisme exemple és
var
    v : enter;

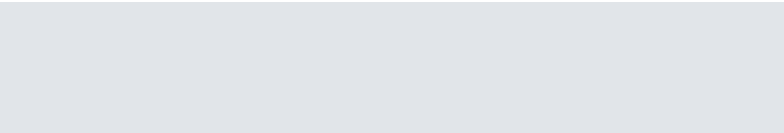
fvar
inici

falgorisme
```

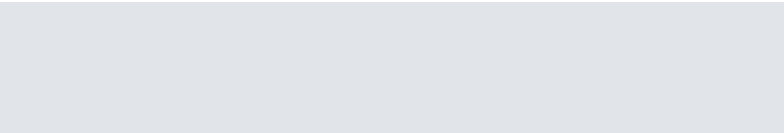
@ 1000



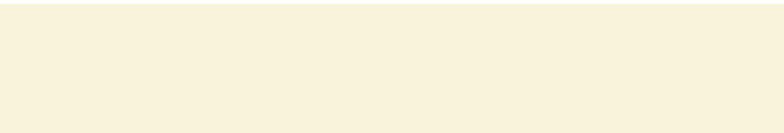
@ 1001



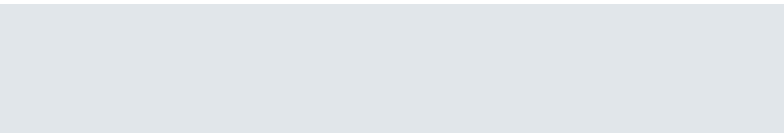
@ 1002



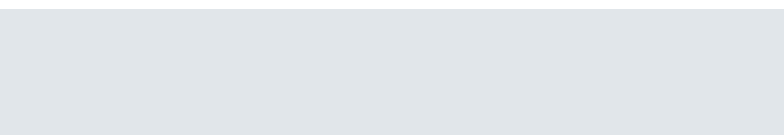
@ 1003



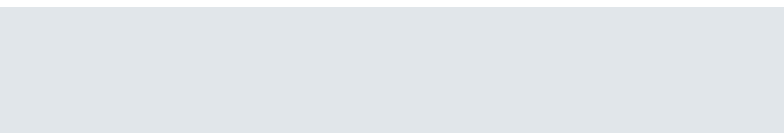
@ 1004



@ 1005



@ 1006



Punters

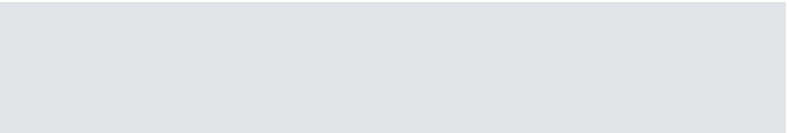
Funcionament

```
algorisme exemple és
var
    v : enter;

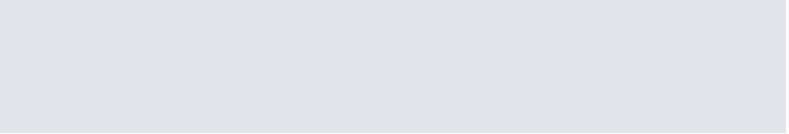
fvar
inici
    v := 200;

falgorisme
```

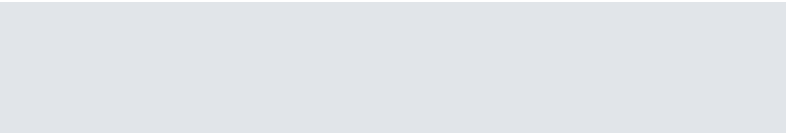
@ 1000



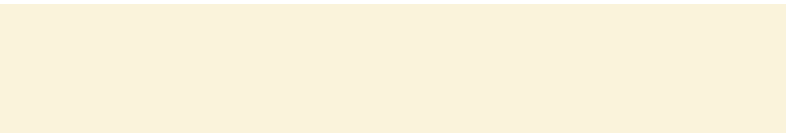
@ 1001



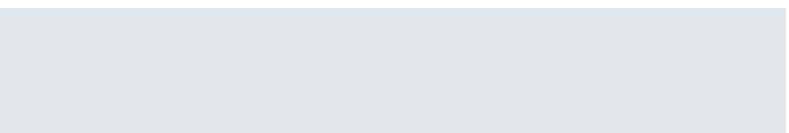
@ 1002



@ 1003



@ 1004



@ 1005



@ 1006



Punters

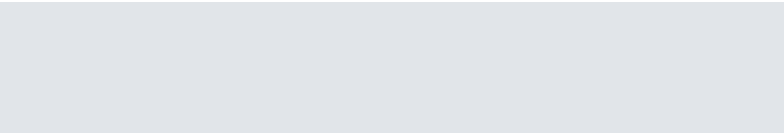
Funcionament

```
algorisme exemple és
var
    v : enter;

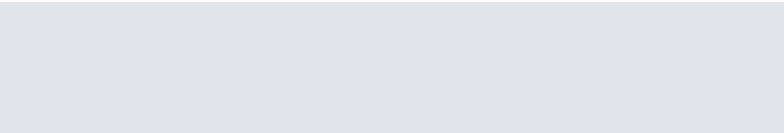
fvar
inici
    v := 200;

falgorisme
```

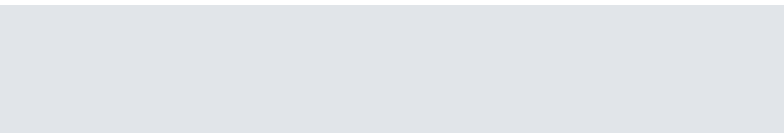
@ 1000



@ 1001



@ 1002



@ 1003



@ 1004



@ 1005



@ 1006



Punters

Funcionament

```
algorisme exemple és
var
  v : enter;

fvar
inici
  v := 200;

falgorisme
```

@ 1000

@ 1001

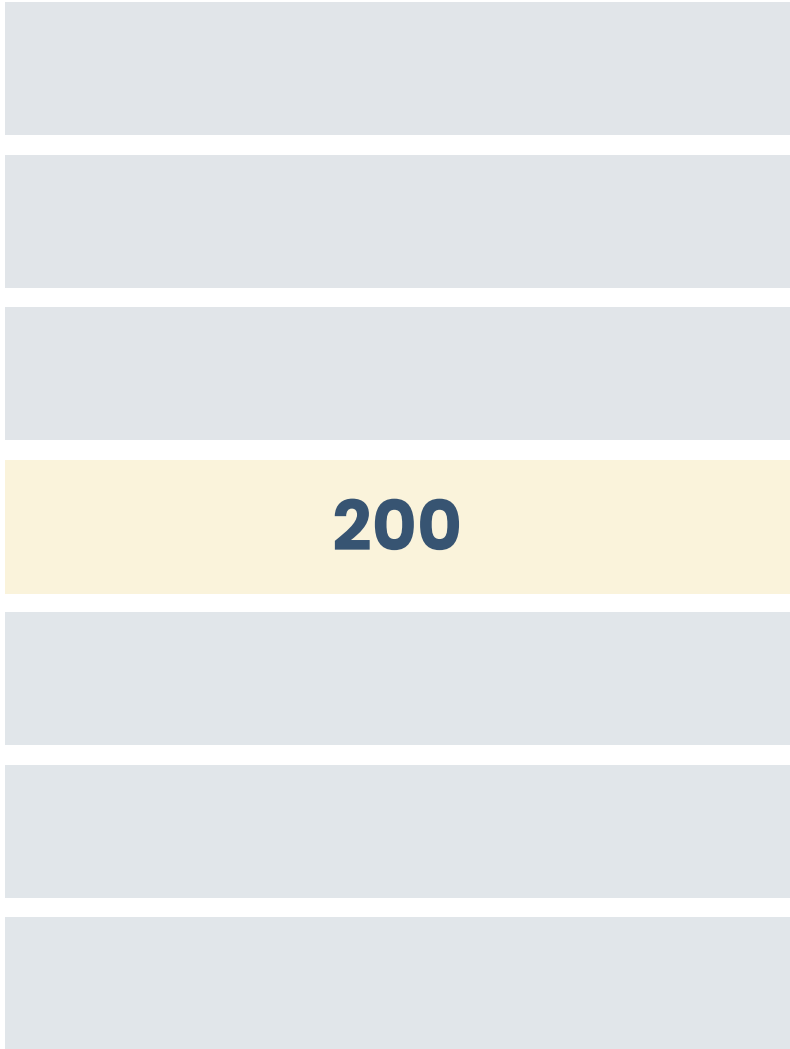
@ 1002

@ 1003

@ 1004

@ 1005

@ 1006



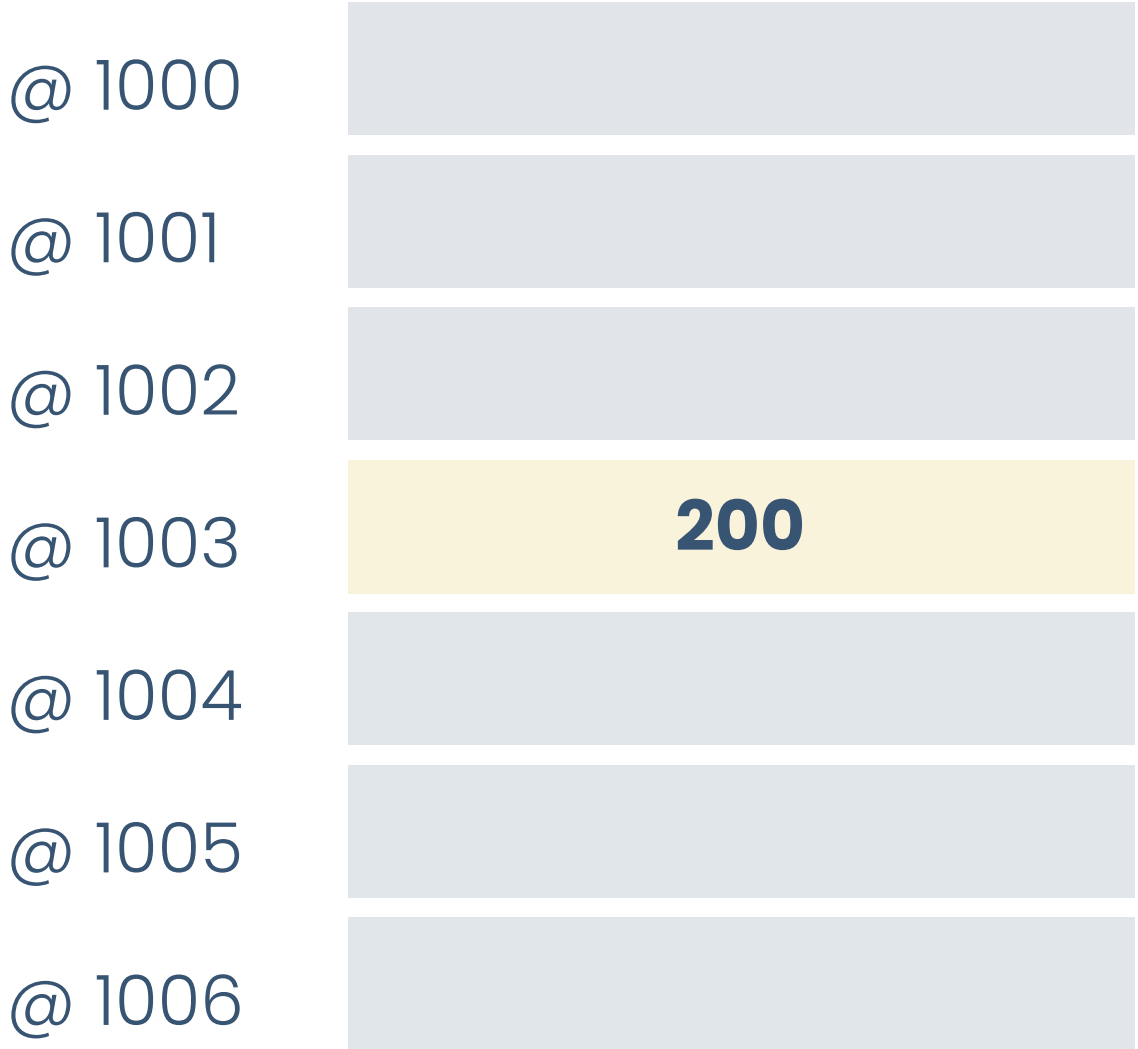
Un punter/apuntador és una variable que conté l'adreça de memòria d'una altra variable.

Punters

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
fvar
inici
  v := 200;

falgorisme
```



Un punter/apuntador és una variable que conté l'adreça de memòria d'una altra variable.

Punters

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
fvar
inici
  v := 200;

falgorisme
```

@ 1000

@ 1001

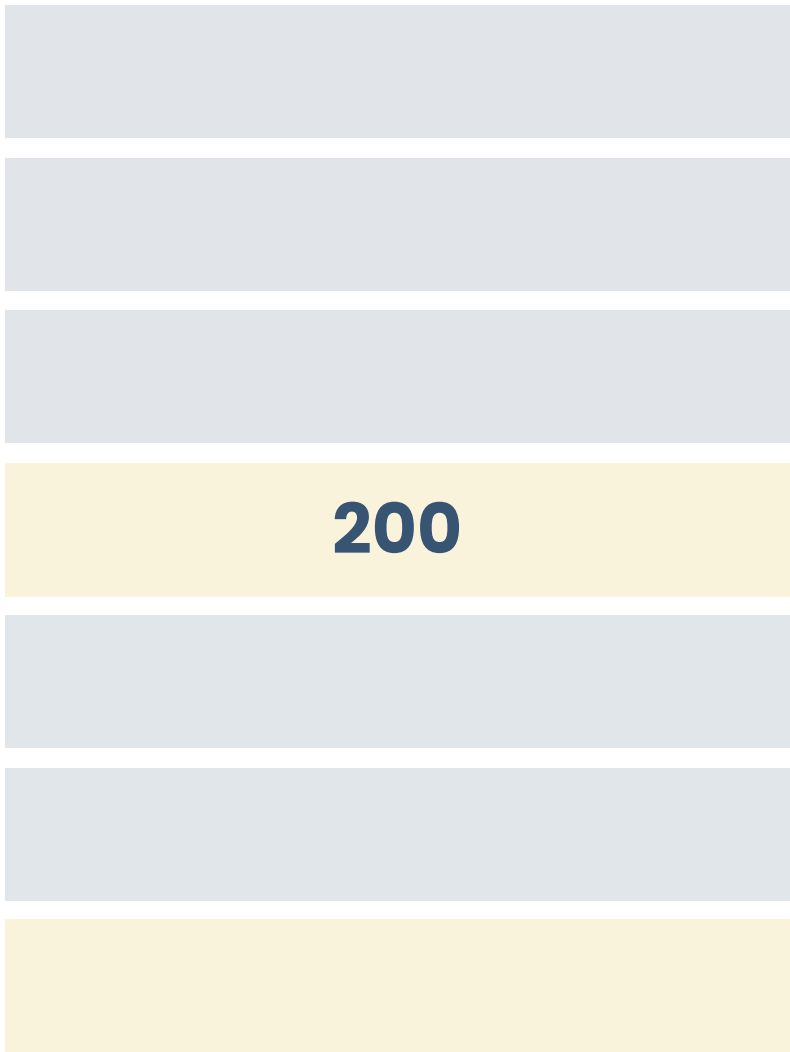
@ 1002

@ 1003

@ 1004

@ 1005

@ 1006



Un punter/apuntador és una variable que conté l'adreça de memòria d'una altra variable.

Punters

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
fvar
inici
  v := 200;

falgorisme
```

@ 1000

@ 1001

@ 1002

@ 1003

@ 1004

@ 1005

@ 1006

200

Un punter/apuntador és una variable que conté l'adreça de memòria d'una altra variable.

- Operador "&": et dóna l'adreça de memòria d'un objecte
- Operador "*": aplicat a un apuntador, dóna accés a l'objecte al qual s'apunta

Punters

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
falgorisme
```

@ 1000

@ 1001

@ 1002

@ 1003

@ 1004

@ 1005

@ 1006

200

Un punter/apuntador és una variable que conté l'adreça de memòria d'una altra variable.

- Operador "&": et dóna l'adreça de memòria d'un objecte
- Operador "*": aplicat a un apuntador, dóna accés a l'objecte al qual s'apunta

Punters

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
falgorisme
```

@ 1000

@ 1001

@ 1002

@ 1003

@ 1004

@ 1005

@ 1006

200

1003

Un punter/apuntador és una variable que conté l'adreça de memòria d'una altra variable.

- Operador "&": et dóna l'adreça de memòria d'un objecte
- Operador "*": aplicat a un apuntador, dóna accés a l'objecte al qual s'apunta

Punters

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
falgorisme
```

@ 1000

@ 1001

@ 1002

@ 1003

@ 1004

@ 1005

@ 1006

200

1003

Un punter/apuntador és una variable que conté l'adreça de memòria d'una altra variable.

- Operador "&": et dóna l'adreça de memòria d'un objecte
- Operador "*": aplicat a un apuntador, dóna accés a l'objecte al qual s'apunta
- En el nostre exemple...
 - **&v** és **l'adreça de la memòria** on es troba el valor de v
----> Adreça 1003
 - ***p** indica que volem **accedir a la dada** que es troba a l'adreça de memòria apuntada pel punter p
----> Què hi ha a l'adreça 1003? El valor 200.

Punters

Ús a teoria

```
algorisme exemple és  
var  
  v : enter;  
  p : punter_a_enter;  
fvar  
inici  
  ...  
falgorisme
```



Els punters es declaren com variables punter_a_tipus, on tipus és qualsevol tipus bàsic o d'usuari. Per exemple:

```
punter_a_enter  
punter_a_caracter  
punter_a_real
```

No té sentit, però, declarar punters a taules, recordeu per què?

@ 1000

@ 1001

@ 1002

@ 1003

@ 1004

@ 1005

@ 1006

200

1003

Punters

Ús a teoria

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
fvar
inici
  ...
falgorisme
```

Els punters es declaren com variables punter_a_tipus, on tipus és qualsevol tipus bàsic o d'usuari. Per exemple:

```
punter_a_enter
punter_a_caracter
punter_a_real
```

No té sentit, però, declarar punters a taules, recordeu per què?

Per obtenir l'adreça de memòria d'un tipus, farem servir l'operador "&".

És correcte fer (seguint el codi anterior): **p = &v;**

Però què ens donaria fer **&p** ?

@ 1000	
@ 1001	
@ 1002	
@ 1003	200
@ 1004	
@ 1005	
@ 1006	1003

Punters

Ús a teoria

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
fvar
inici
  ...
falgorisme
```

Els punters es declaren com variables punter_a_tipus, on tipus és qualsevol tipus bàsic o d'usuari. Per exemple:

```
punter_a_enter
punter_a_caracter
punter_a_real
```

No té sentit, però, declarar punters a taules, recordeu per què?

Per obtenir l'adreça de memòria d'un tipus, farem servir l'operador "&".

És correcte fer (seguint el codi anterior): **p = &v;**

Però què ens donaria fer **&p** ?

Ens donaria l'adreça de p (1006)

@ 1000	
@ 1001	
@ 1002	
@ 1003	200
@ 1004	
@ 1005	
@ 1006	1003

Punters

Ús a teoria

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
fvar
inici
  ...
falgorisme
```

Els punters es declaren com variables punter_a_tipus, on tipus és qualsevol tipus bàsic o d'usuari. Per exemple:

```
punter_a_enter
punter_a_caracter
punter_a_real
```

No té sentit, però, declarar punters a taules, recordeu per què?

Per obtenir l'adreça de memòria d'un tipus, farem servir l'operador "&".

És correcte fer (seguint el codi anterior): **p = &v;**

Però què ens donaria fer **&p** ?

Ens donaria l'adreça de p (1006)

Per accedir al contingut d'un punter, farem servir l'operador "*"

És correcte fer (seguint el codi anterior): ***p**

Però què ens donaria fer ***v** ?

@ 1000	
@ 1001	
@ 1002	
@ 1003	200
@ 1004	
@ 1005	
@ 1006	1003

Punters

Ús a teoria

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
fvar
inici
  ...
falgorisme
```

Els punters es declaren com variables punter_a_tipus, on tipus és qualsevol tipus bàsic o d'usuari. Per exemple:

```
punter_a_enter
punter_a_caracter
punter_a_real
```

No té sentit, però, declarar punters a taules, recordeu per què?

Per obtenir l'adreça de memòria d'un tipus, farem servir l'operador "&".

És correcte fer (seguint el codi anterior): **p = &v;**

Però què ens donaria fer **&p** ?

Ens donaria l'adreça de p (1006)

Per accedir al contingut d'un punter, farem servir l'operador "*"

És correcte fer (seguint el codi anterior): ***p**

Però què ens donaria fer ***v** ?

Ens donaria un error, no es pot accedir als continguts d'on apunta v perquè "v" no és un apuntador

@ 1000

@ 1001

@ 1002

@ 1003

@ 1004

@ 1005

@ 1006

200

1003

Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és  
var
```

```
fvar  
inici
```

```
falgorisme
```

@ 1000

@ 1001

@ 1002

@ 1003

@ 1004

@ 1005

@ 1006

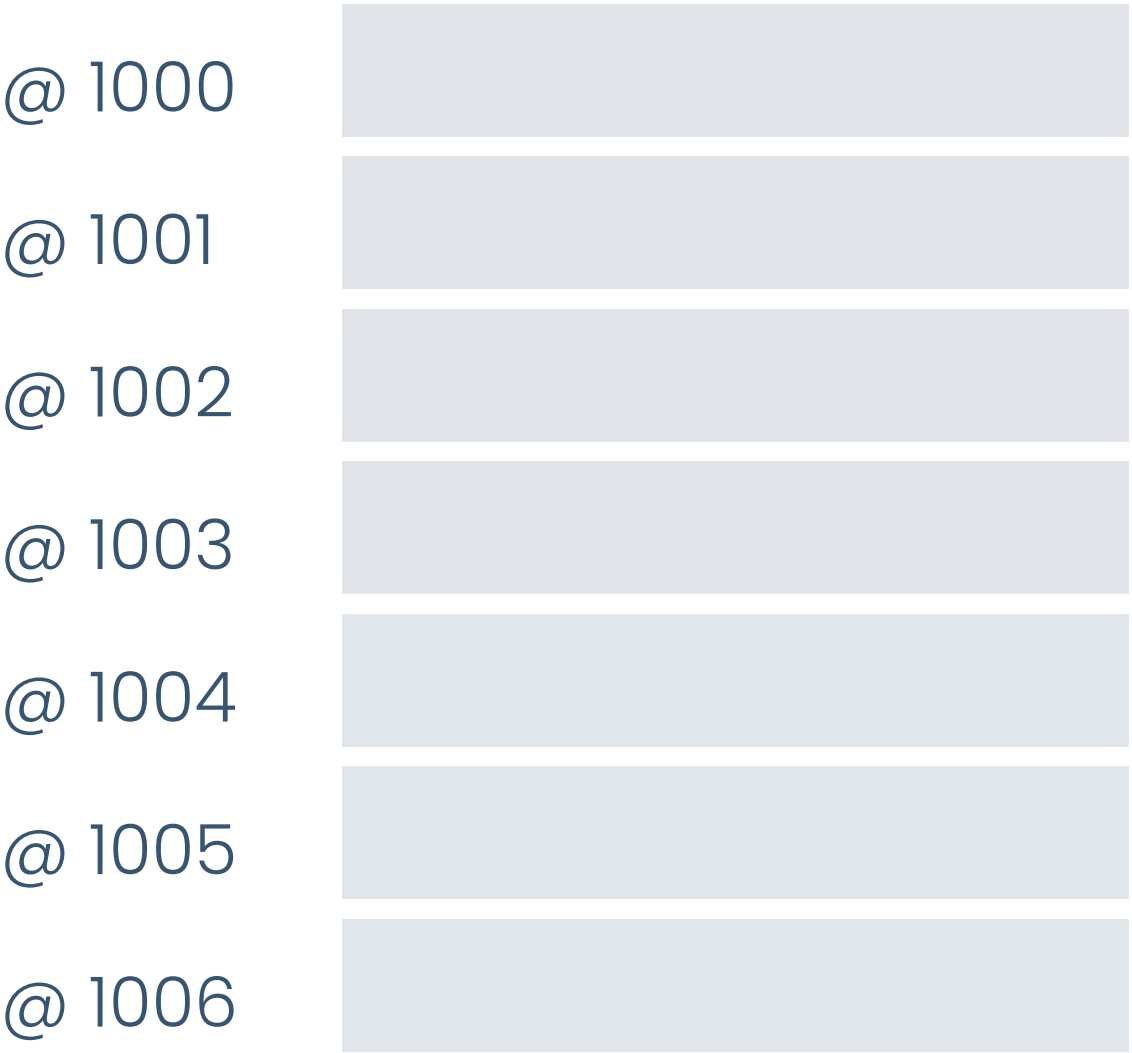
Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;

fvar
inici

falgorisme
```



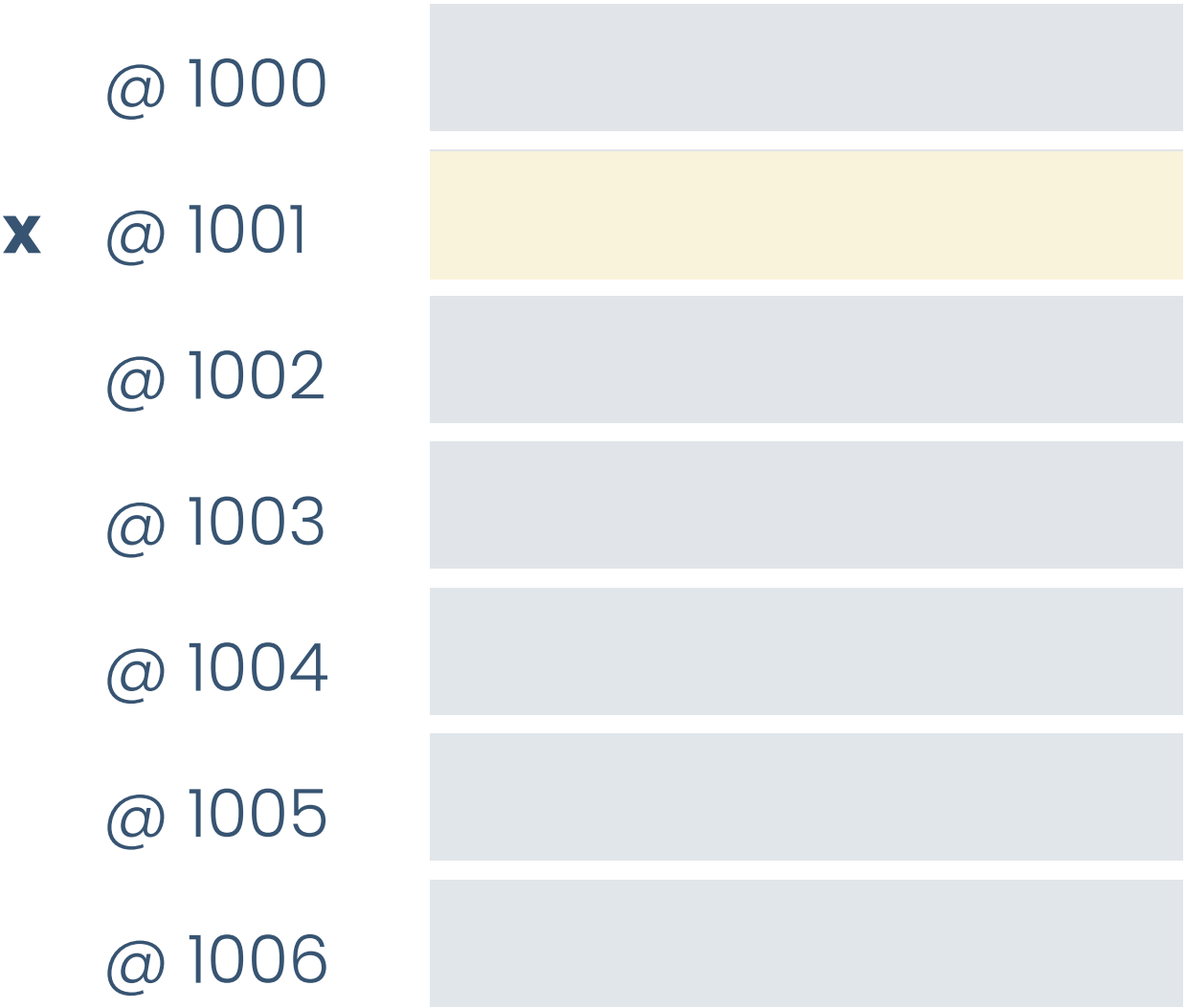
Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;

fvar
inici

falgorisme
```



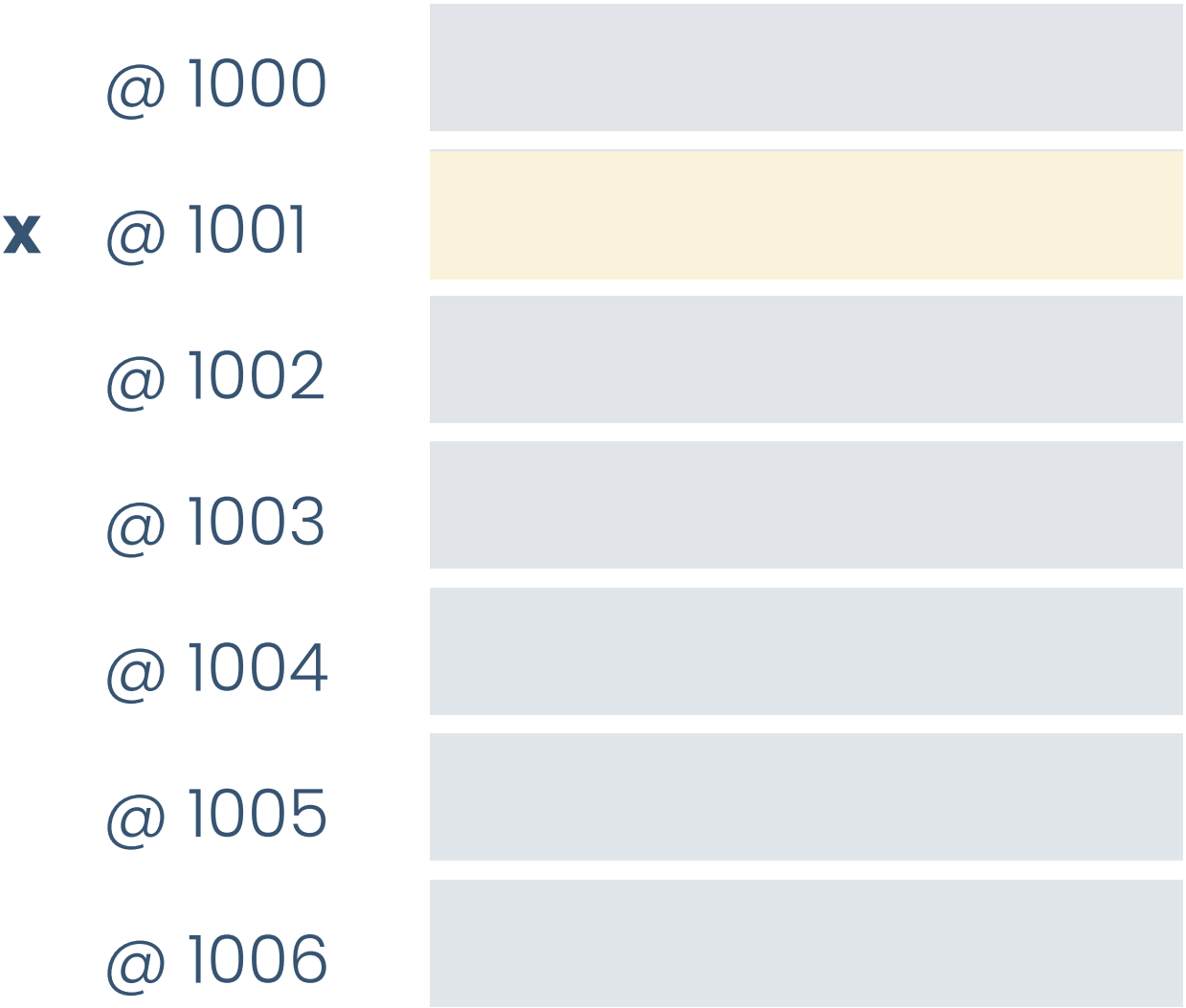
Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;

fvar
inici

falgorisme
```



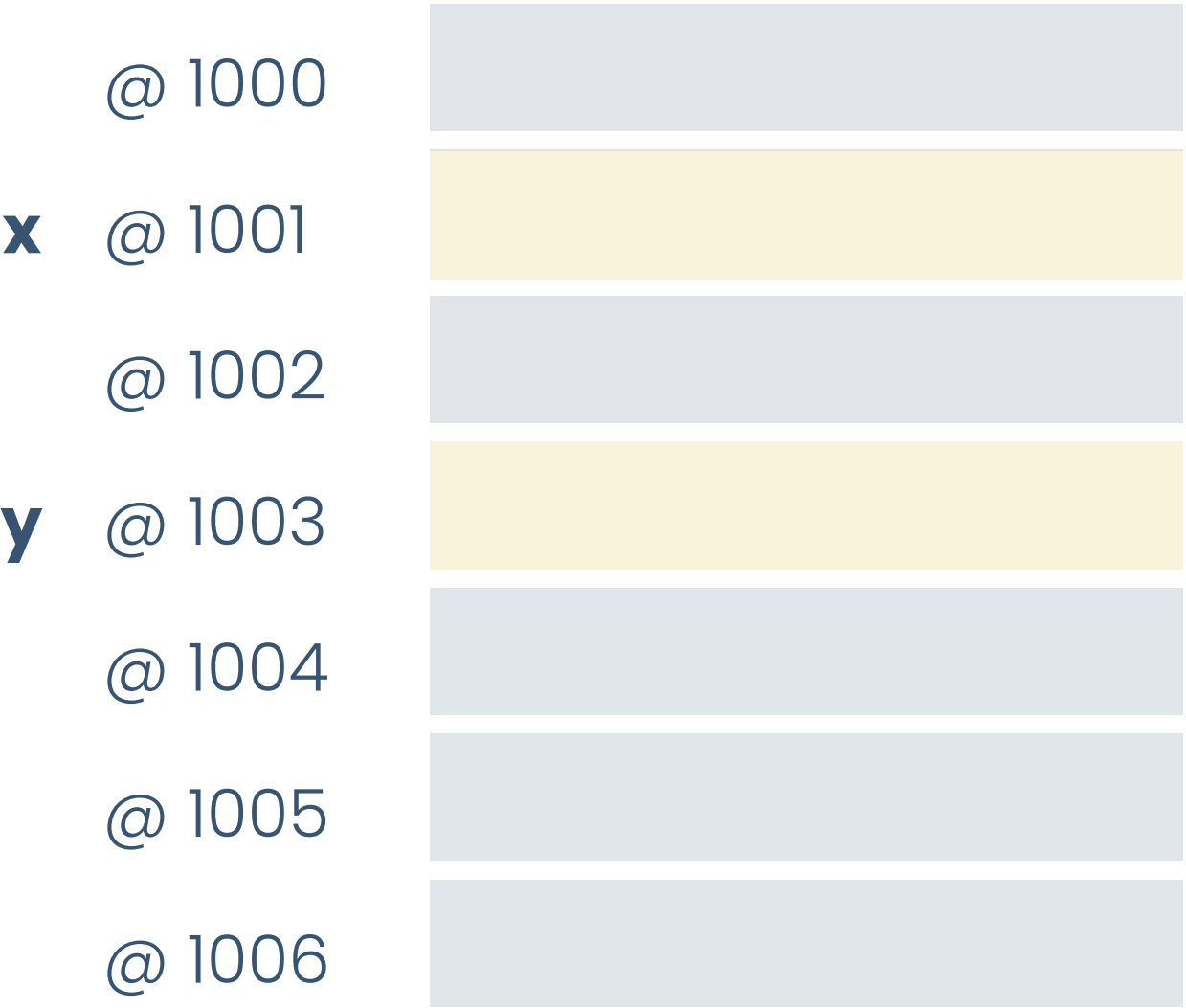
Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;

fvar
inici

falgorisme
```

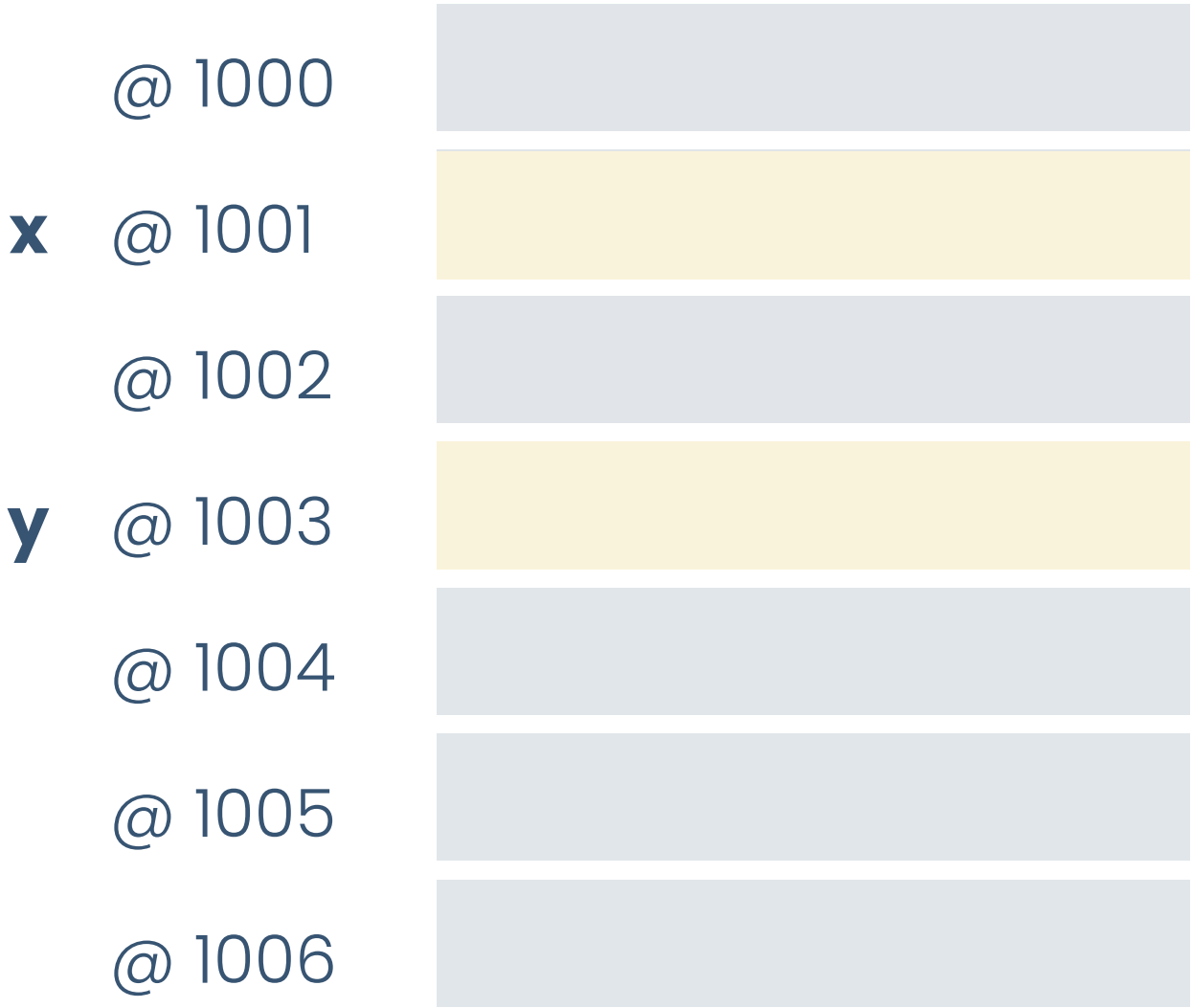


Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
```

falgorisme

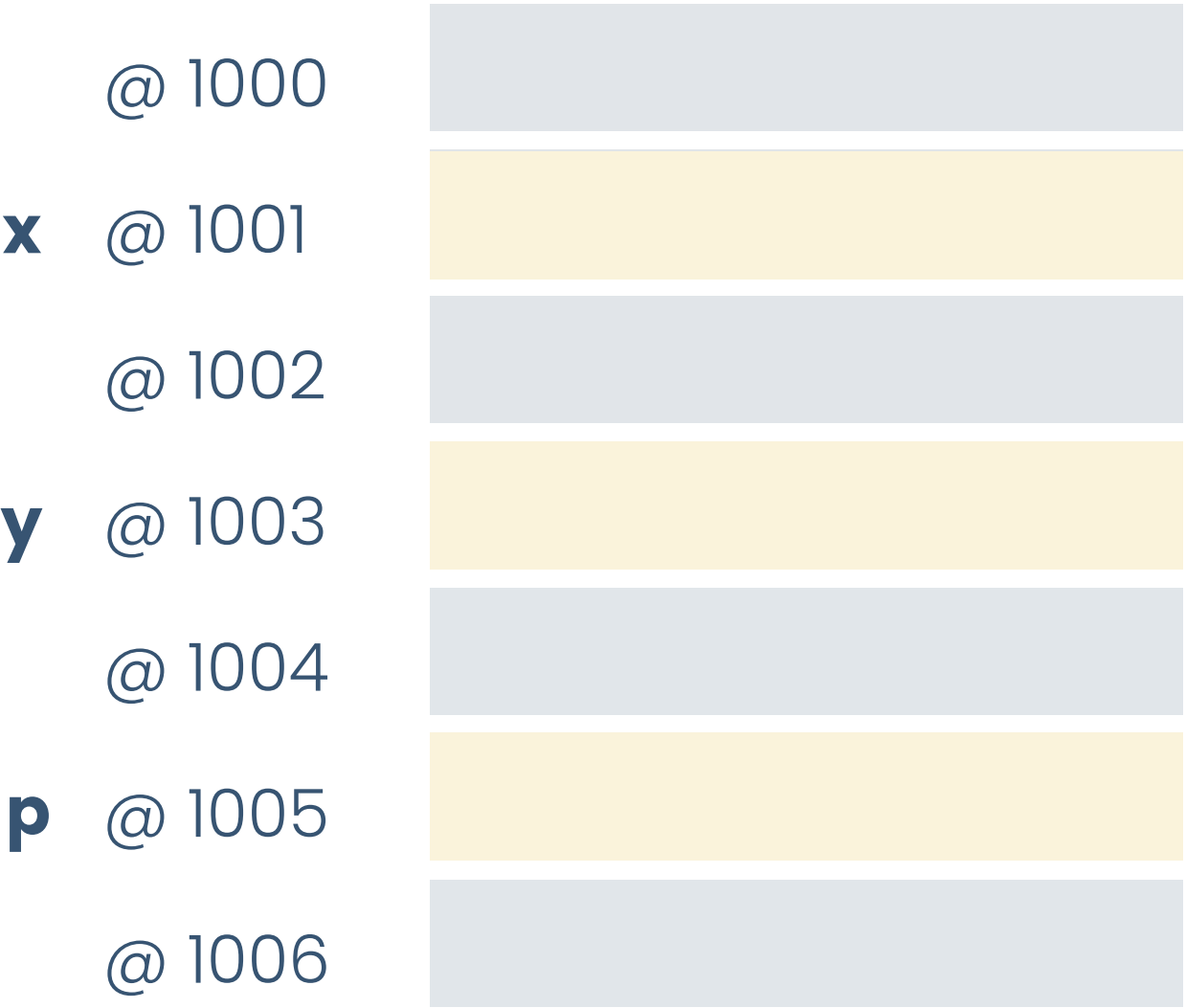


Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici

falgorisme
```



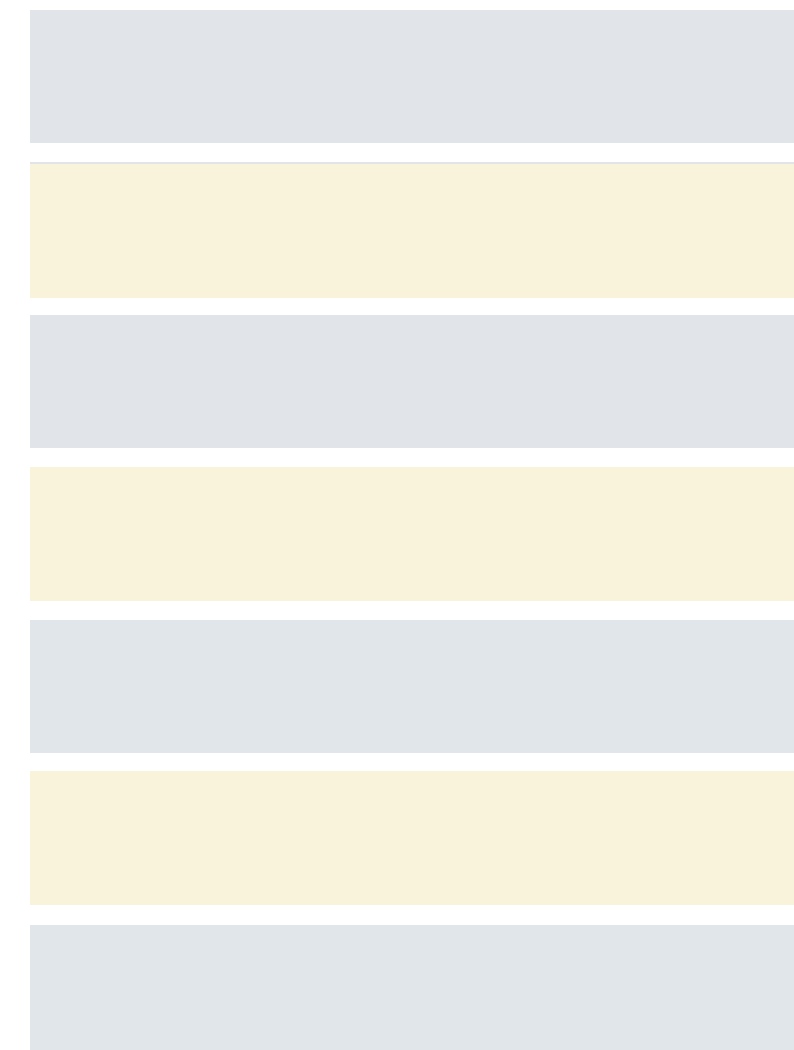
Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;

falgorisme
```

	@ 1000	
x	@ 1001	
	@ 1002	
y	@ 1003	
	@ 1004	
p	@ 1005	
	@ 1006	

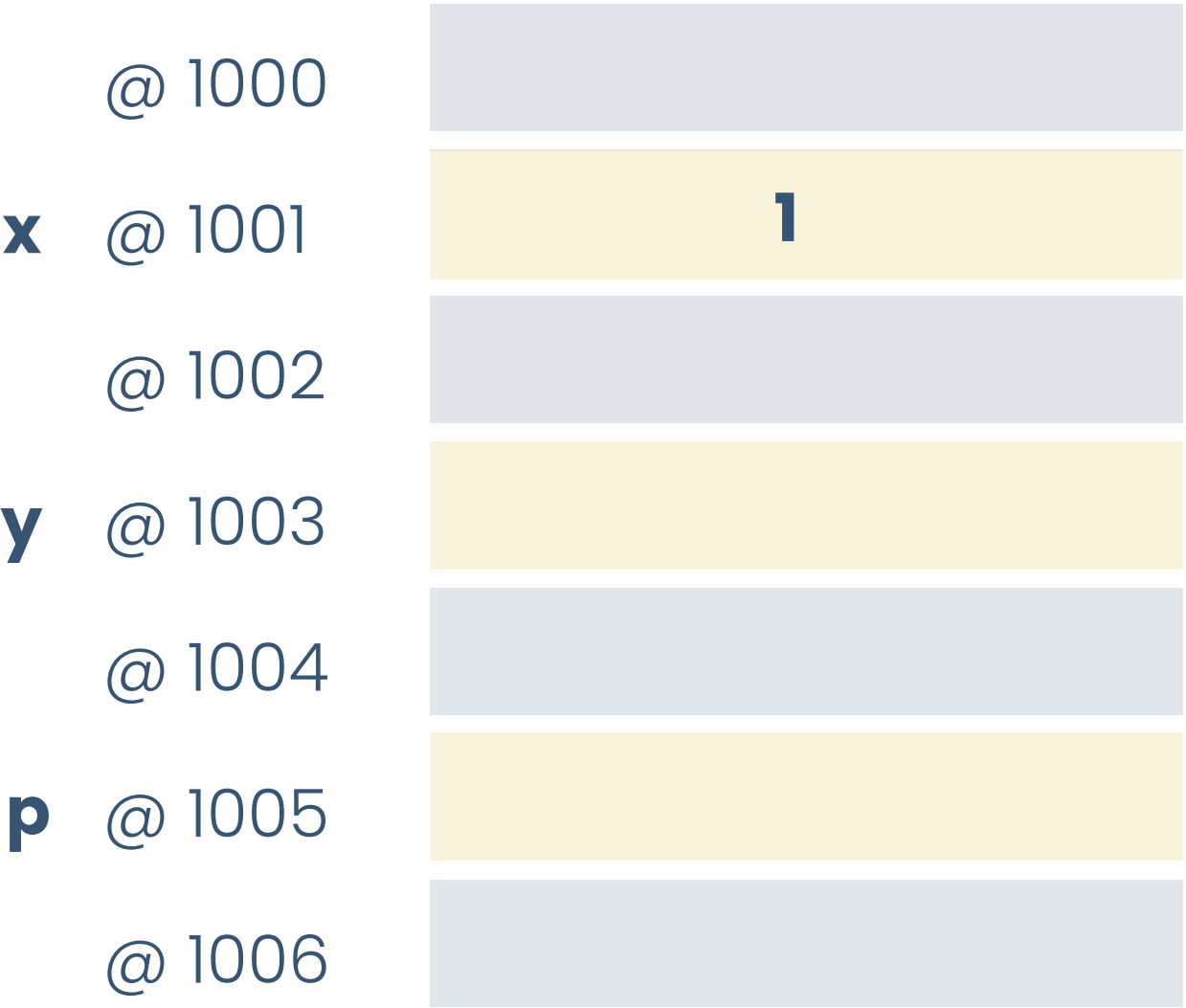


Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;

falgorisme
```

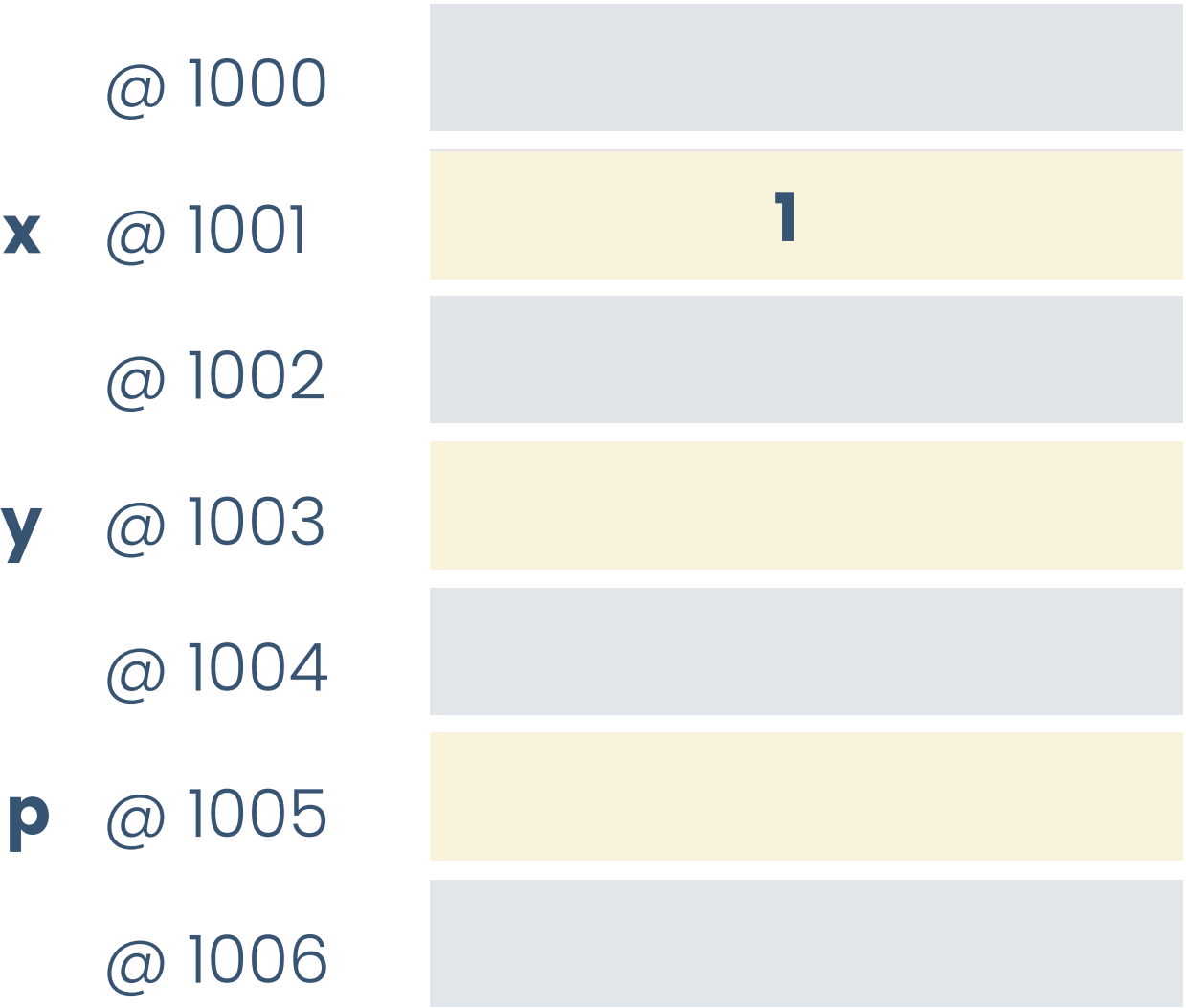


Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;
  y := 2;

falgorisme
```



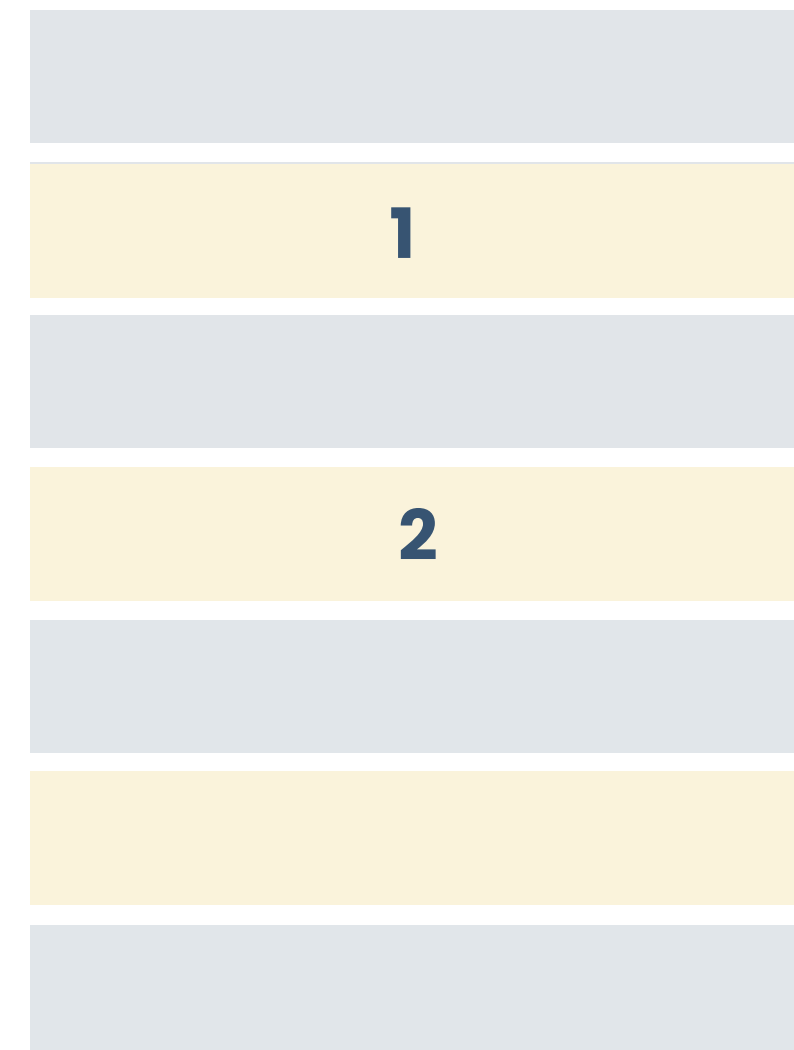
Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;
  y := 2;

falgorisme
```

	@ 1000
x	@ 1001
	@ 1002
y	@ 1003
	@ 1004
p	@ 1005
	@ 1006



Punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
```

```
var
```

```
  x : enter;
```

```
  y : enter;
```

```
  p : punter_a_enter;
```

```
fvar
```

```
inici
```

```
  x := 1;
```

```
  y := 2;
```

```
falgorisme
```

@ 1000

x @ 1001

@ 1002

y @ 1003

@ 1004

p @ 1005

@ 1006

1

2

Punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
```

```
var
```

```
  x : enter;
```

```
  y : enter;
```

```
  p : punter_a_enter;
```

```
fvar
```

```
inici
```

```
  x := 1;
```

```
  y := 2;
```

```
  p := &x;
```

```
falgorisme
```

@ 1000

x @ 1001

@ 1002

y @ 1003

@ 1004

p @ 1005

@ 1006

1

2

Punters

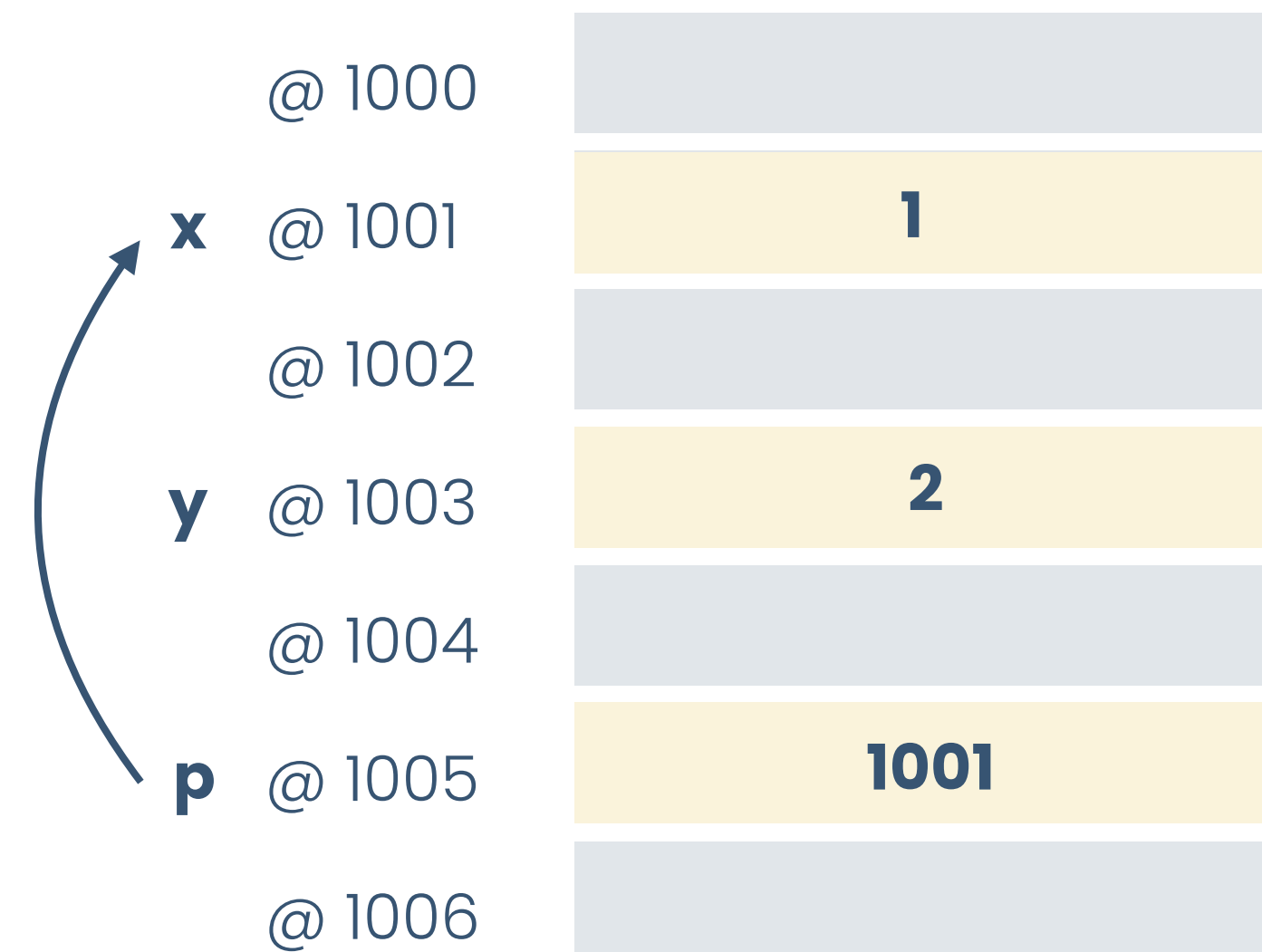
Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;
  y := 2;
  p := &x; // p apunta a x
```

falgorisme

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta



Punters

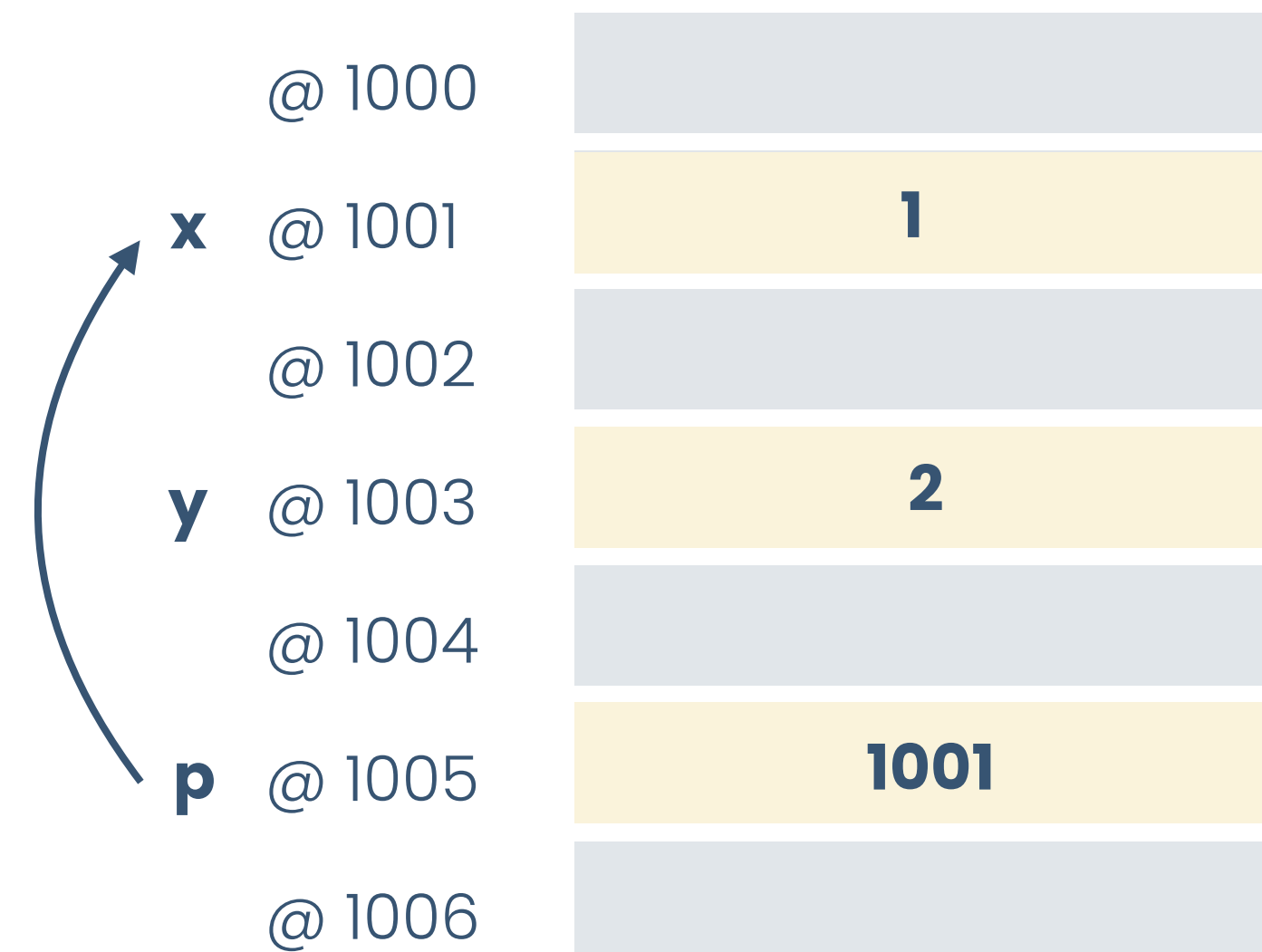
Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;
  y := 2;
  p := &x; // p apunta a x
  y := *p;
```

falgorisme

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta



Punters

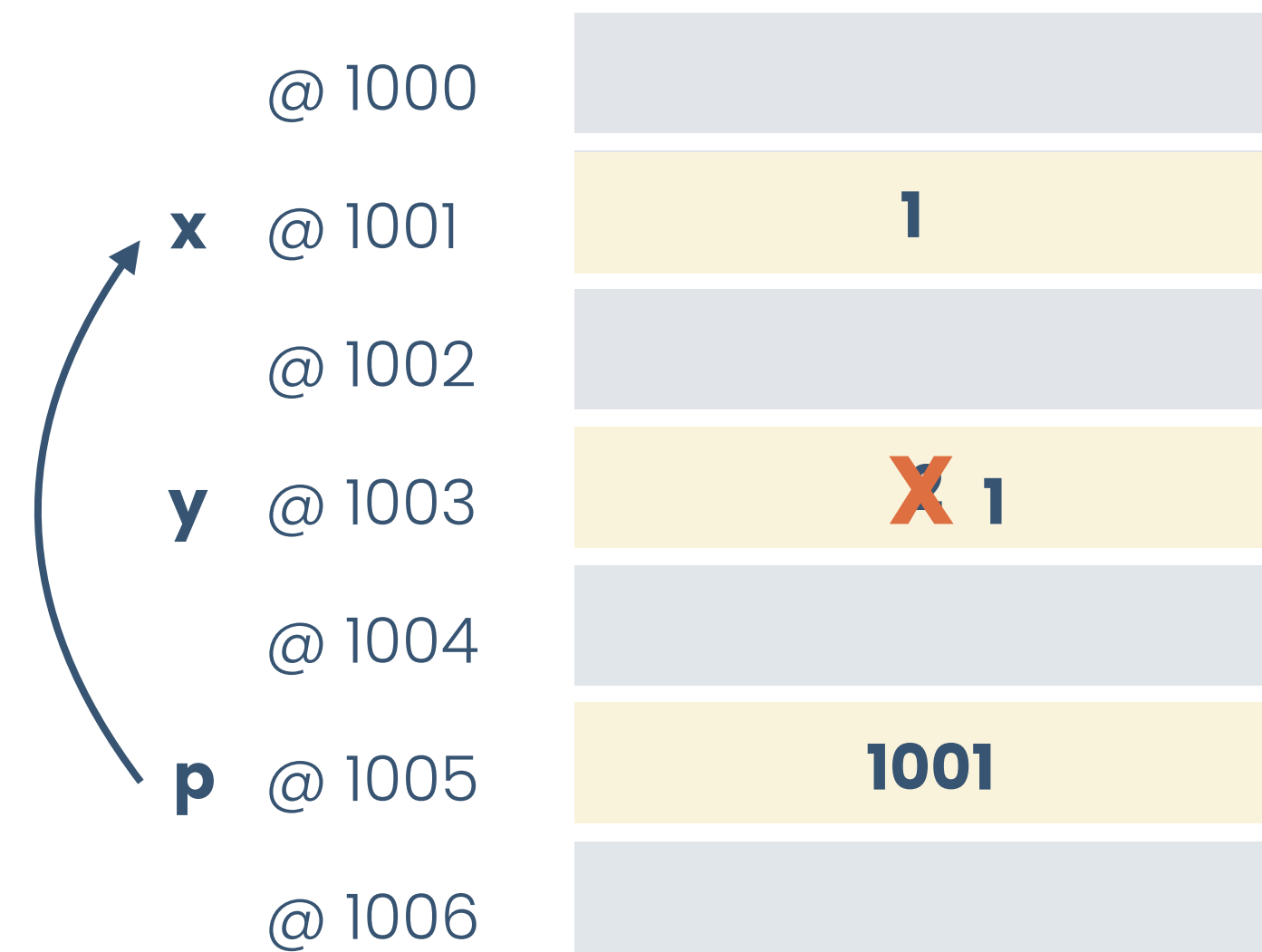
Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;
  y := 2;
  p := &x; // p apunta a x
  y := *p; // y val 1
```

falgorisme

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta



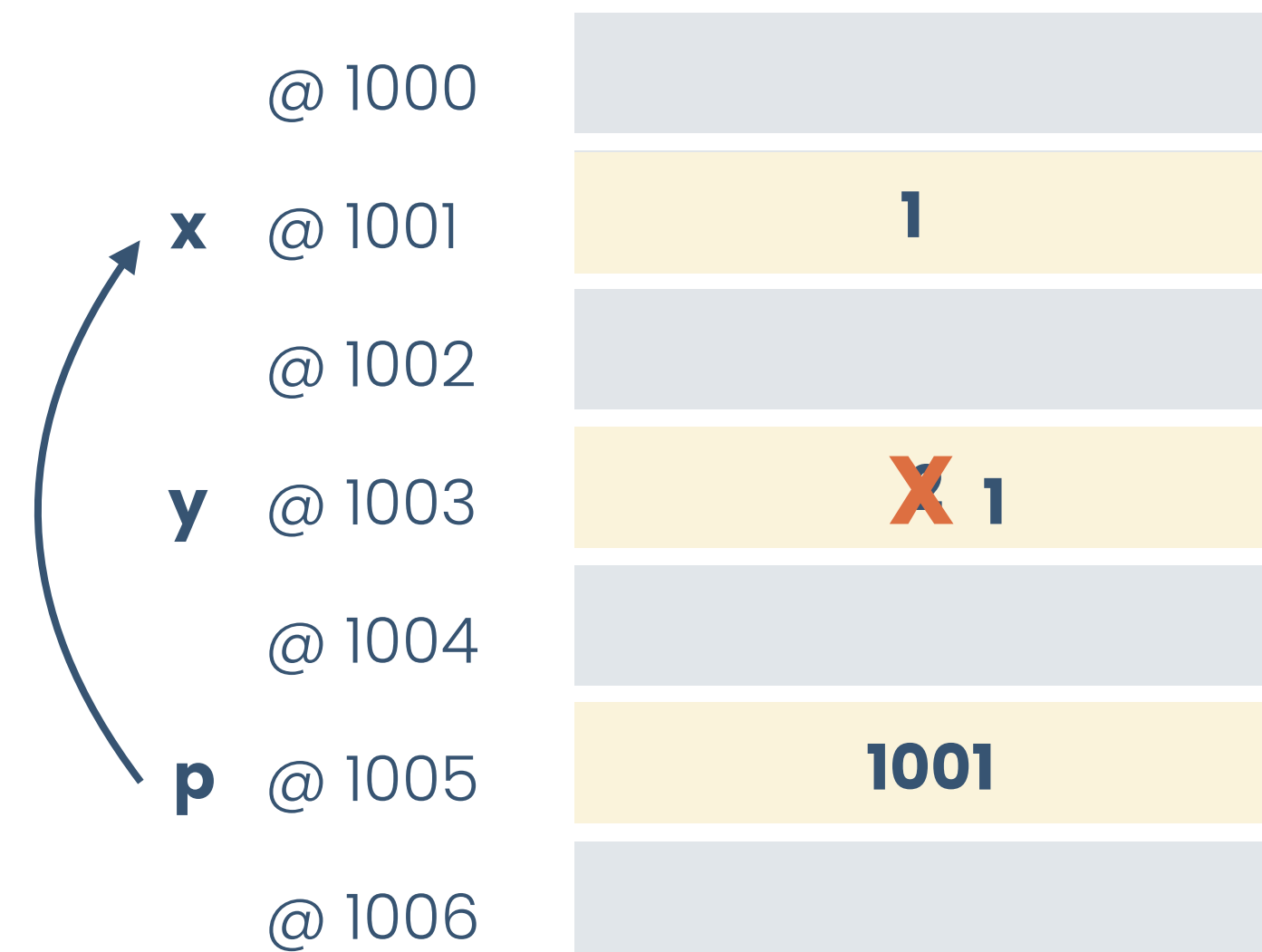
Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;
  y := 2;
  p := &x; // p apunta a x
  y := *p; // y val 1
  *p := 0;

falgorisme
```

Operador "&": et dóna l'adreça de memòria d'un objecte
Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

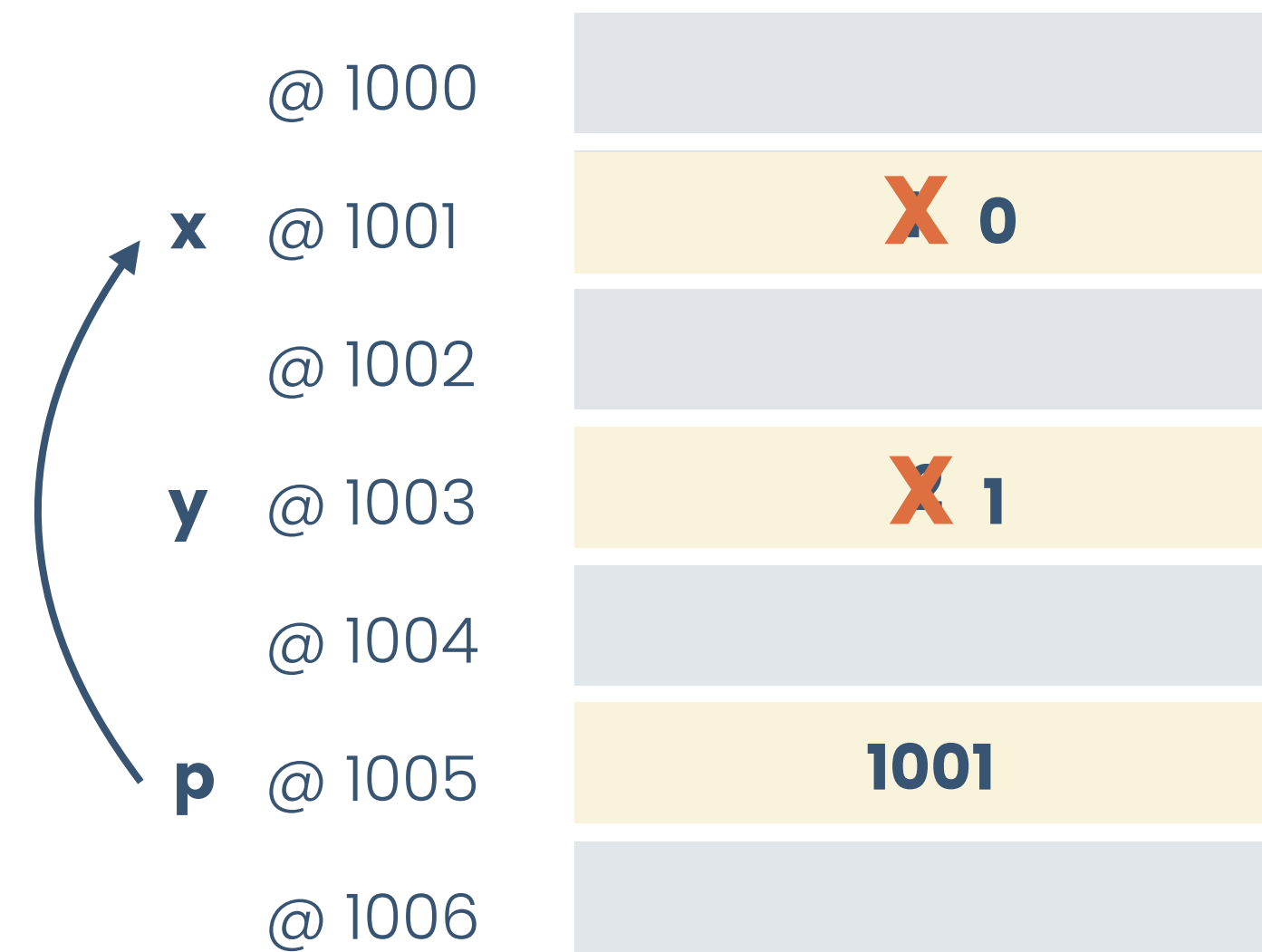


Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;
  y := 2;
  p := &x; // p apunta a x
  y := *p; // y val 1
  *p := 0; // x val 0
falgorisme
```

Operador "&": et dóna l'adreça de memòria d'un objecte
Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta



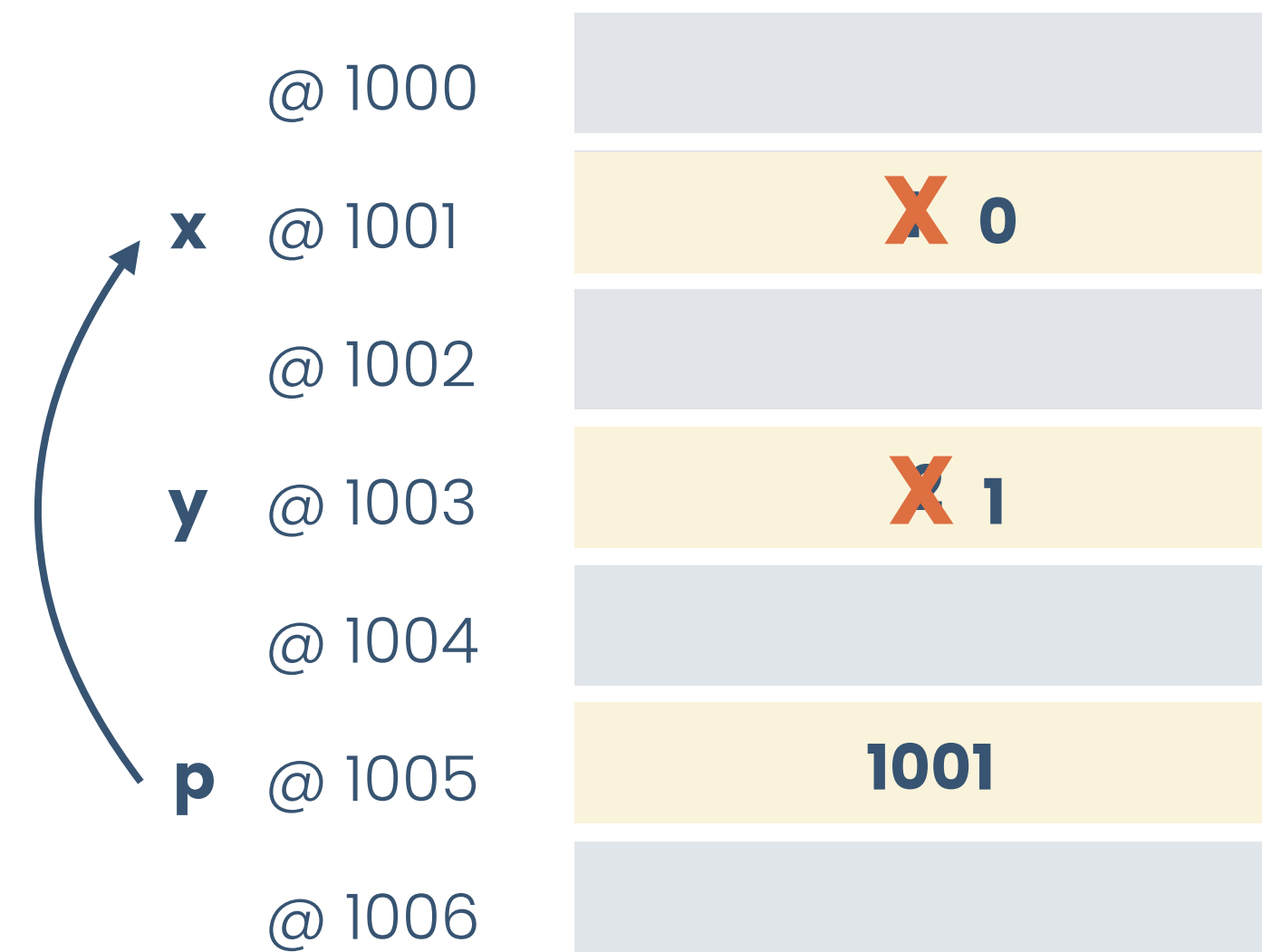
Punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Funcionament: exemple pas a pas (II)

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;
  y := 2;
  p := &x; // p apunta a x
  y := *p; // y val 1
  *p := 0; // x val 0
falgorisme
```



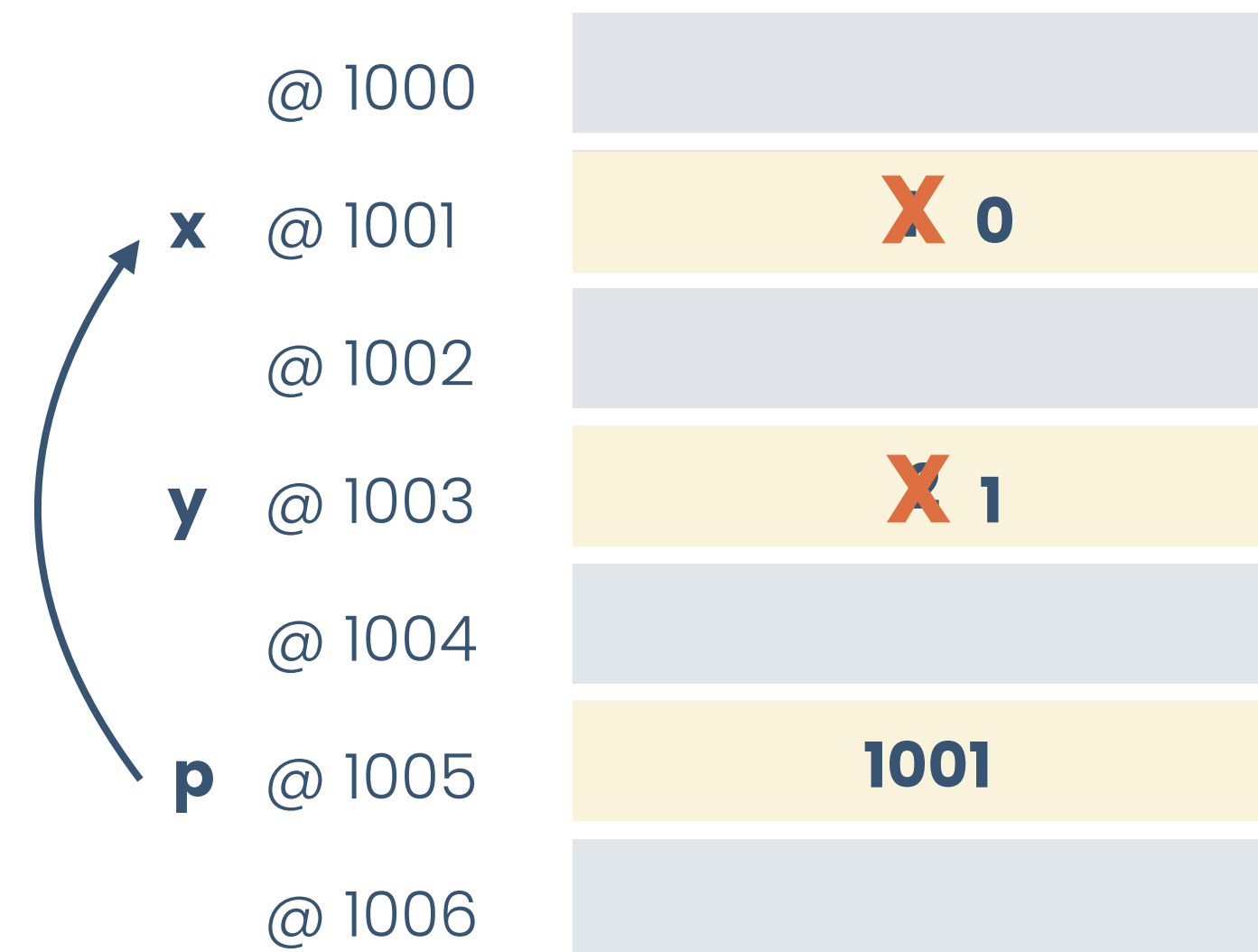
Punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Funcionament: exemple pas a pas (II)

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;
  y := 2;
  p := &x; // p apunta a x
  y := *p; // y val 1
  *p := 0; // x val 0
  p := &y;
falgorisme
```

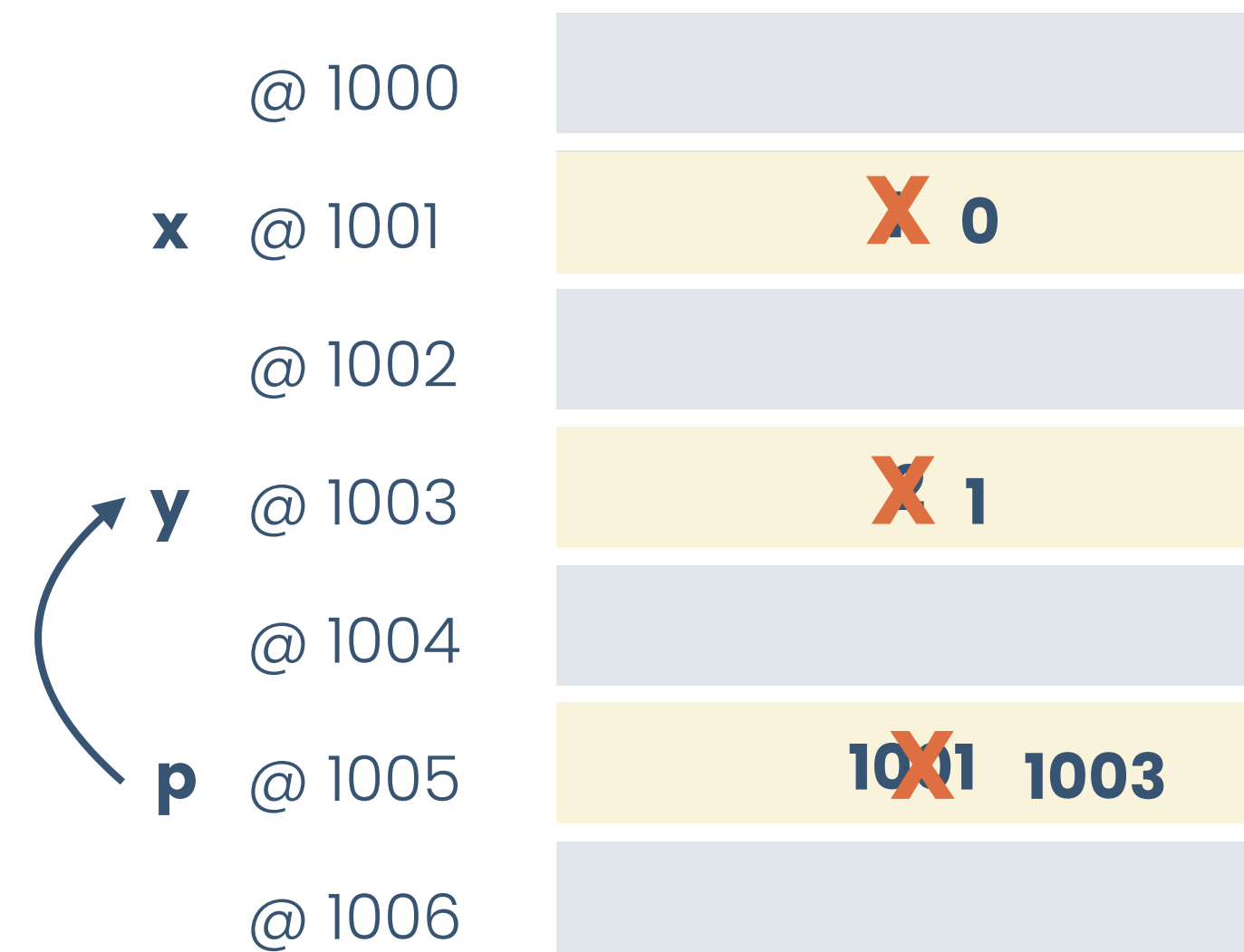


Punters

Funcionament: exemple pas a pas (II)

```
algorisme exemple_2 és
var
  x : enter;
  y : enter;
  p : punter_a_enter;
fvar
inici
  x := 1;
  y := 2;
  p := &x; // p apunta a x
  y := *p; // y val 1
  *p := 0; // x val 0
  p := &y; // p apunta a y
falgorisme
```

Operador "&": et dóna l'adreça de memòria d'un objecte
Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

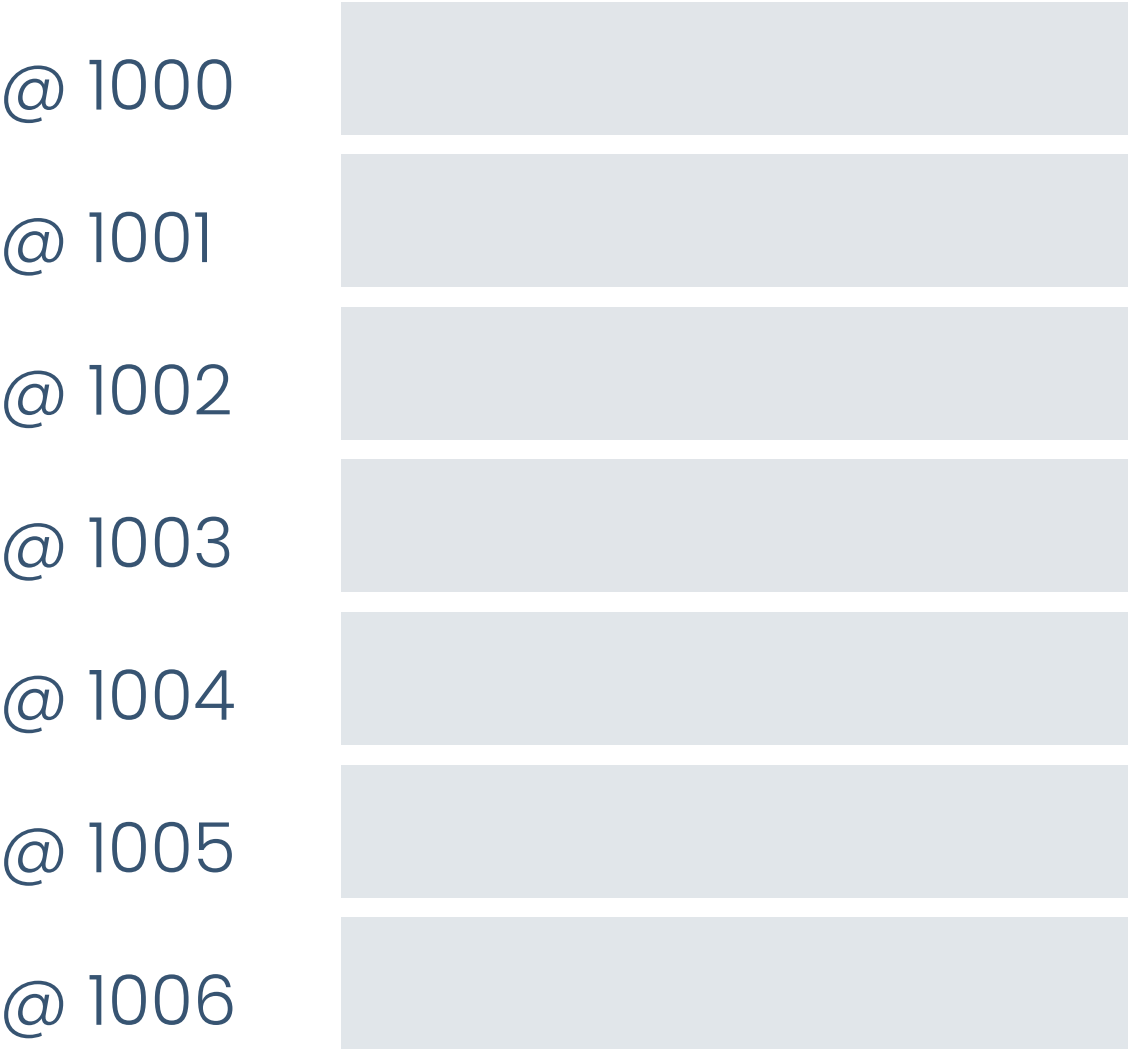


Punters

Assignar punters a altres punters

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  q : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;

falgorisme
```

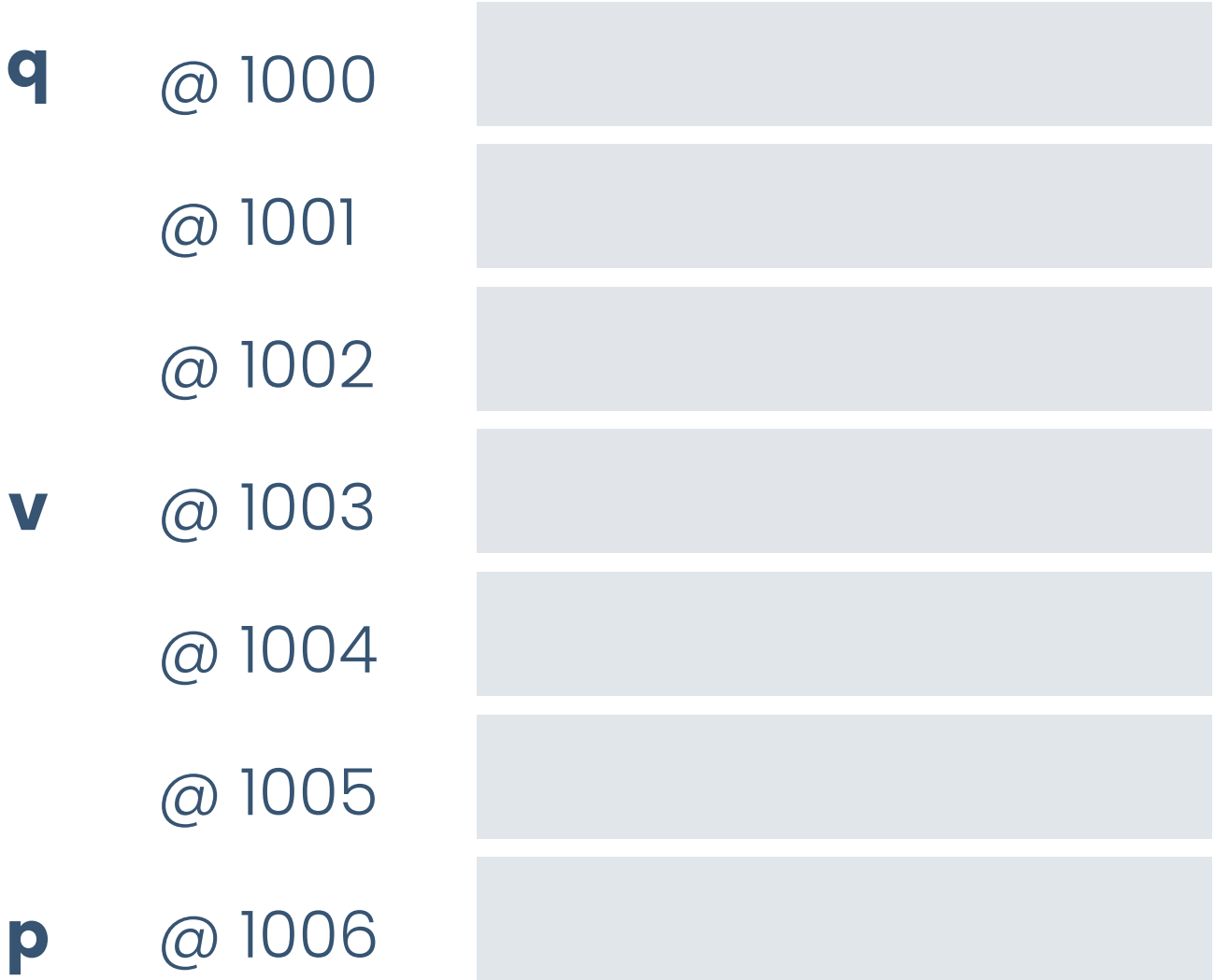


Punters

Assignar punters a altres punters

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  q : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;

falgorisme
```

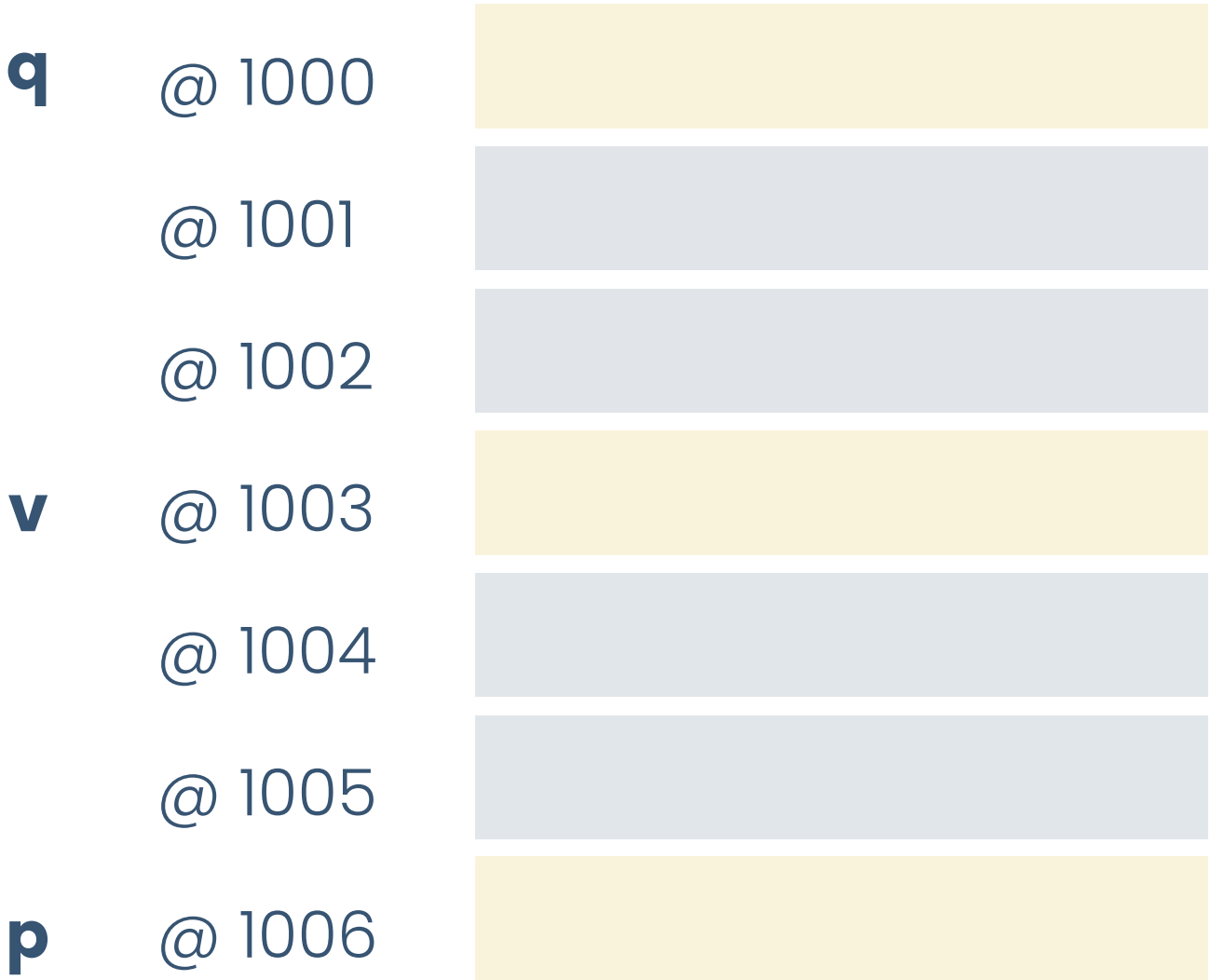


Punters

Assignar punters a altres punters

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  q : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;

falgorisme
```



Punters

Assignar punters a altres punters

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  q : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;

falgorisme
```

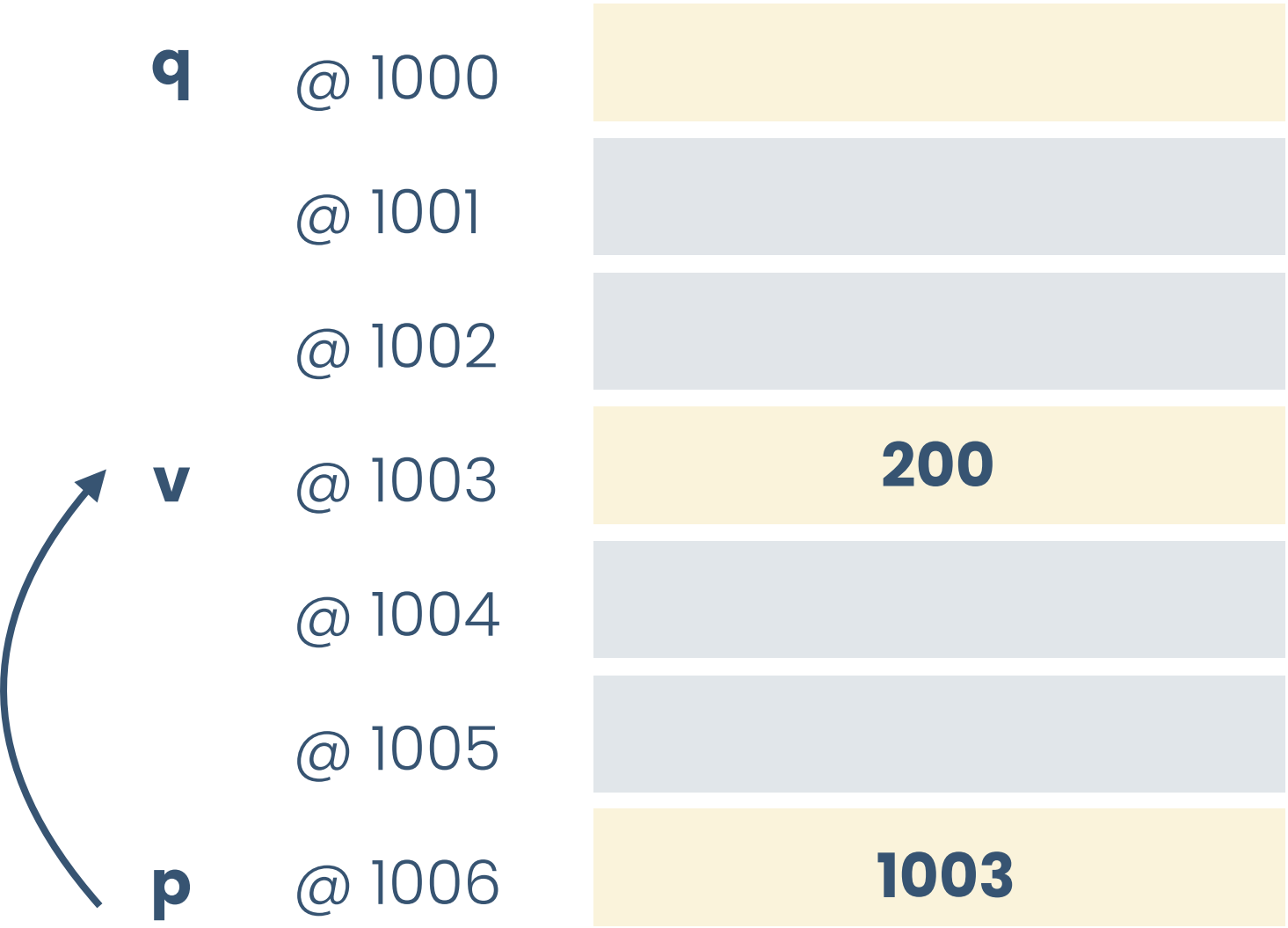


Punters

Assignar punters a altres punters

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  q : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;

falgorisme
```

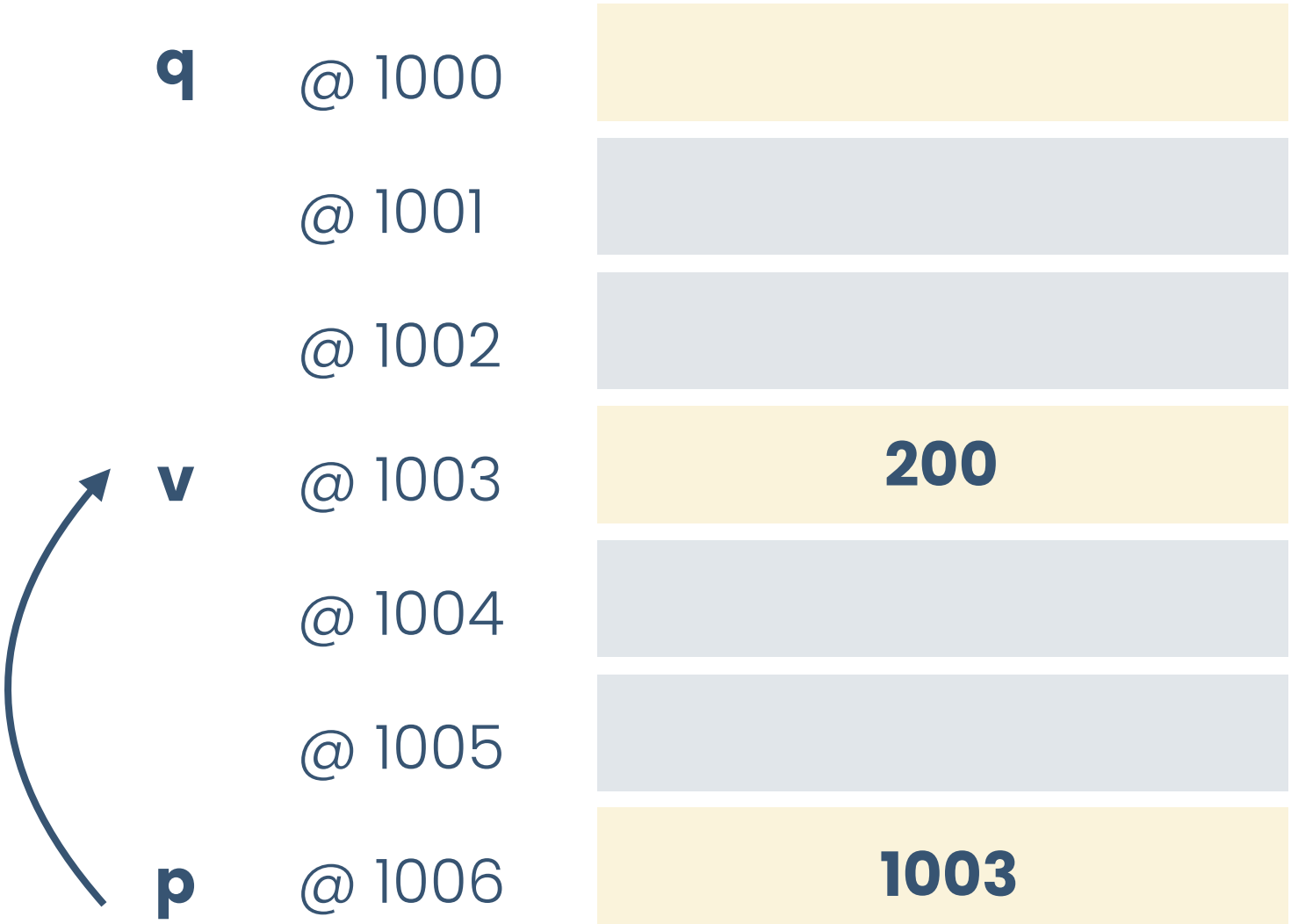


Punters

Assignar punters a altres punters

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  q : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
  q := p;
falgorisme
```

→ Què fa aquesta línia?

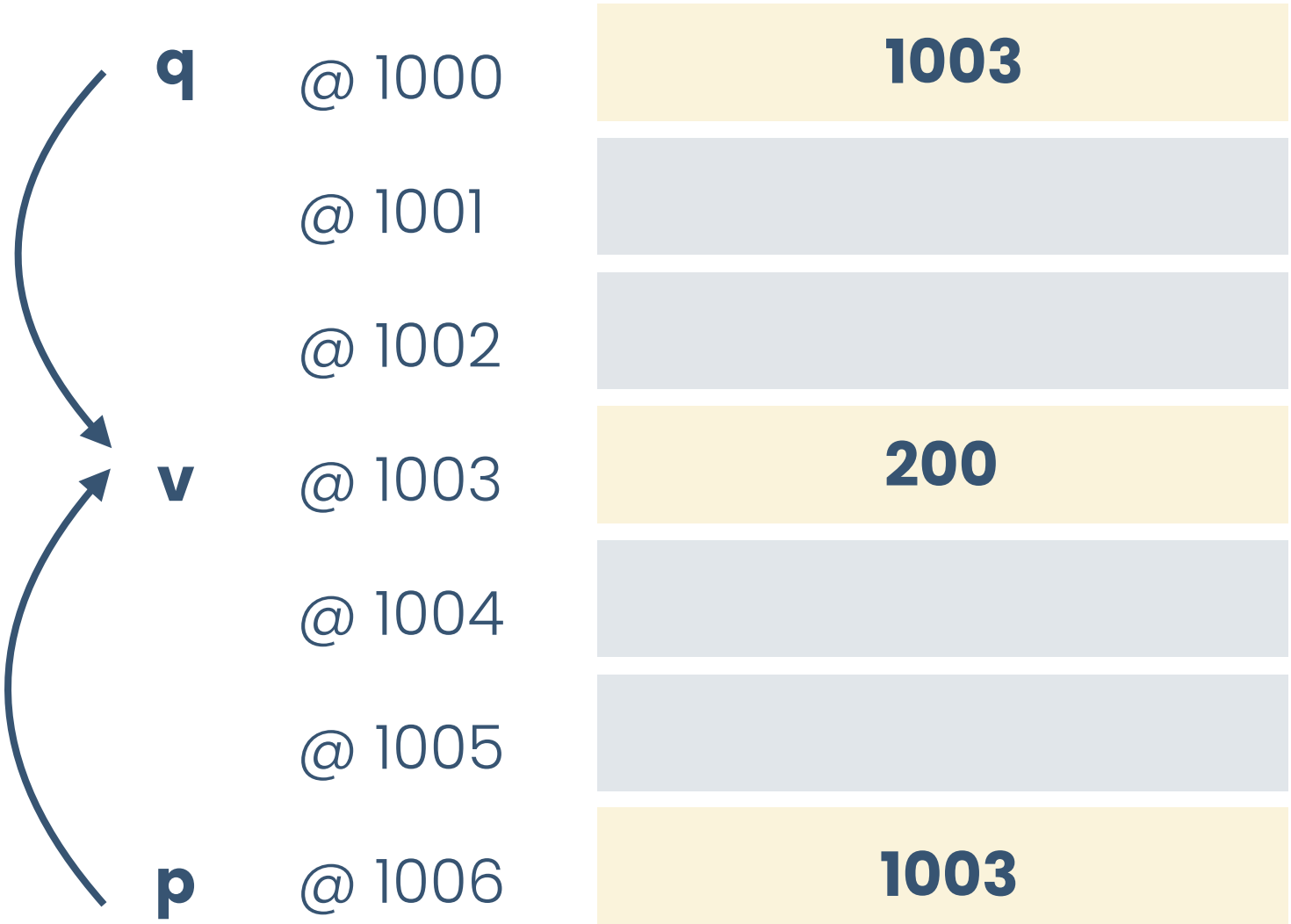


Punters

Assignar punters a altres punters

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  q : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
  q := p;
falgorisme
```

Què fa aquesta línia?

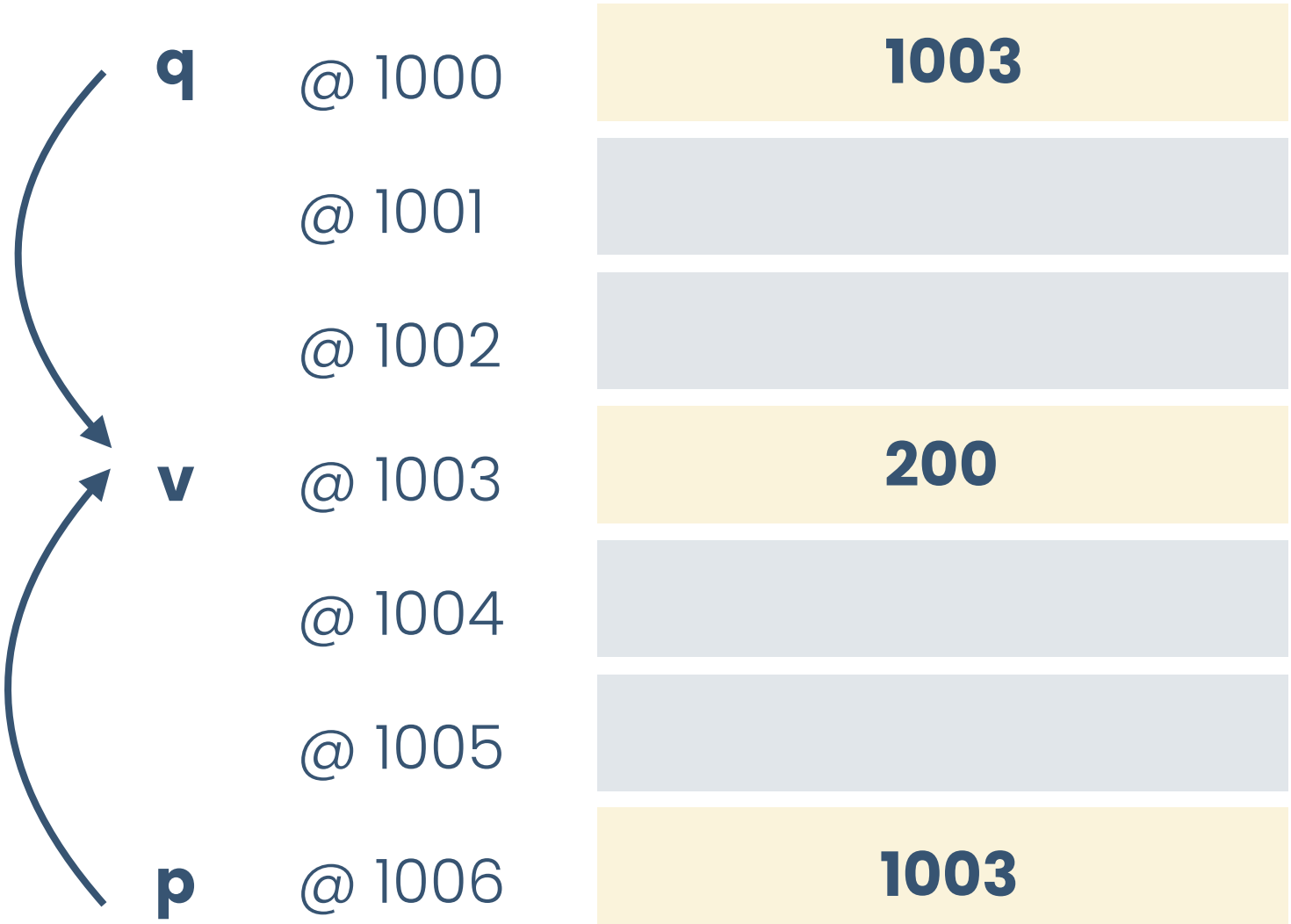


Punters

Assignar punters a altres punters

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  q : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
  q := p;
  *p := 300;
falgorisme
```

Què fa aquesta línia?

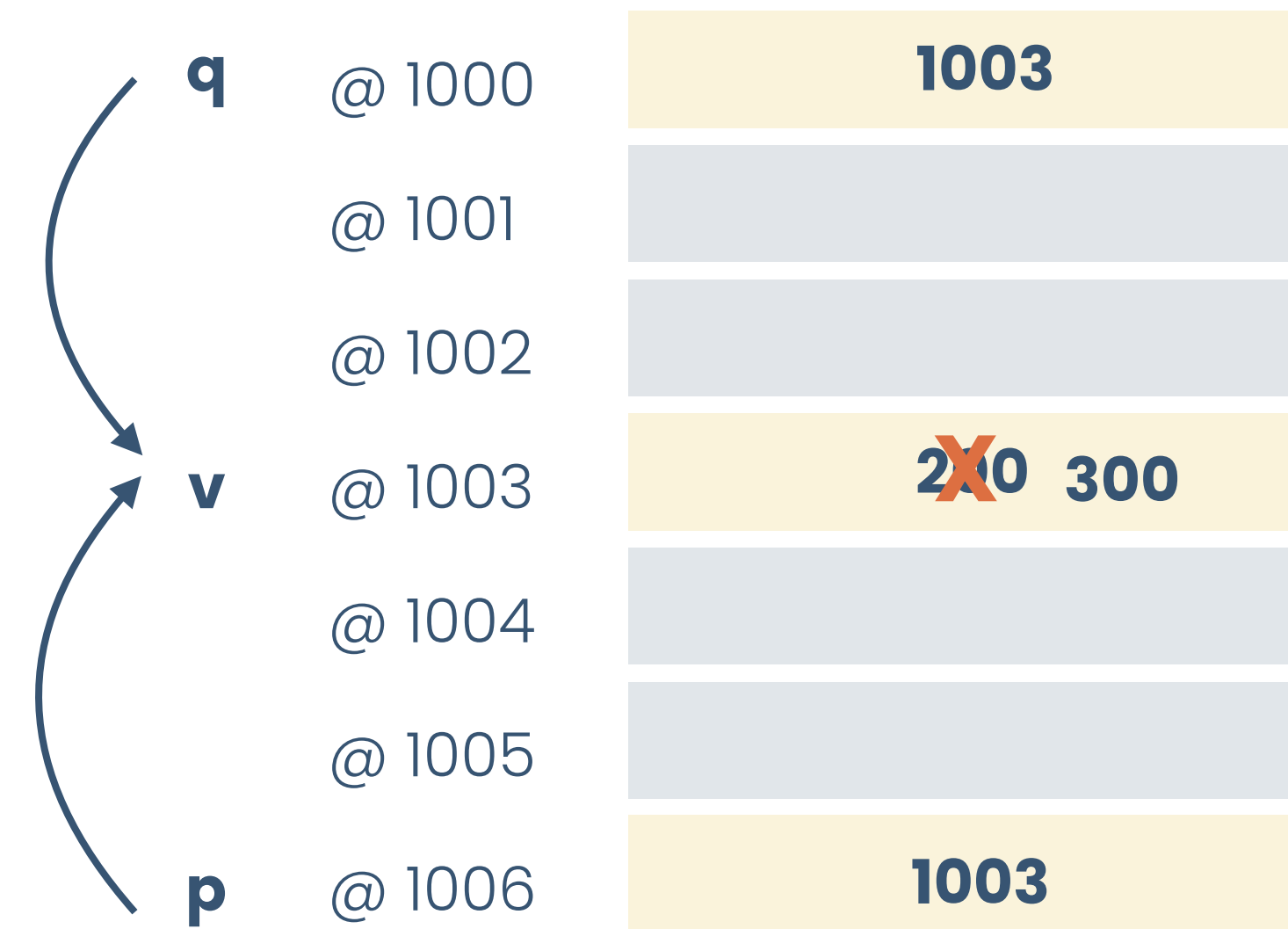


Punters

Assignar punters a altres punters

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  q : punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
  q := p;
  *p := 300;
falgorisme
```

Què fa aquesta línia?



Punters

Operador "&": et dóna l'adreça de memòria d'un objecte
Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Exercici per practicar

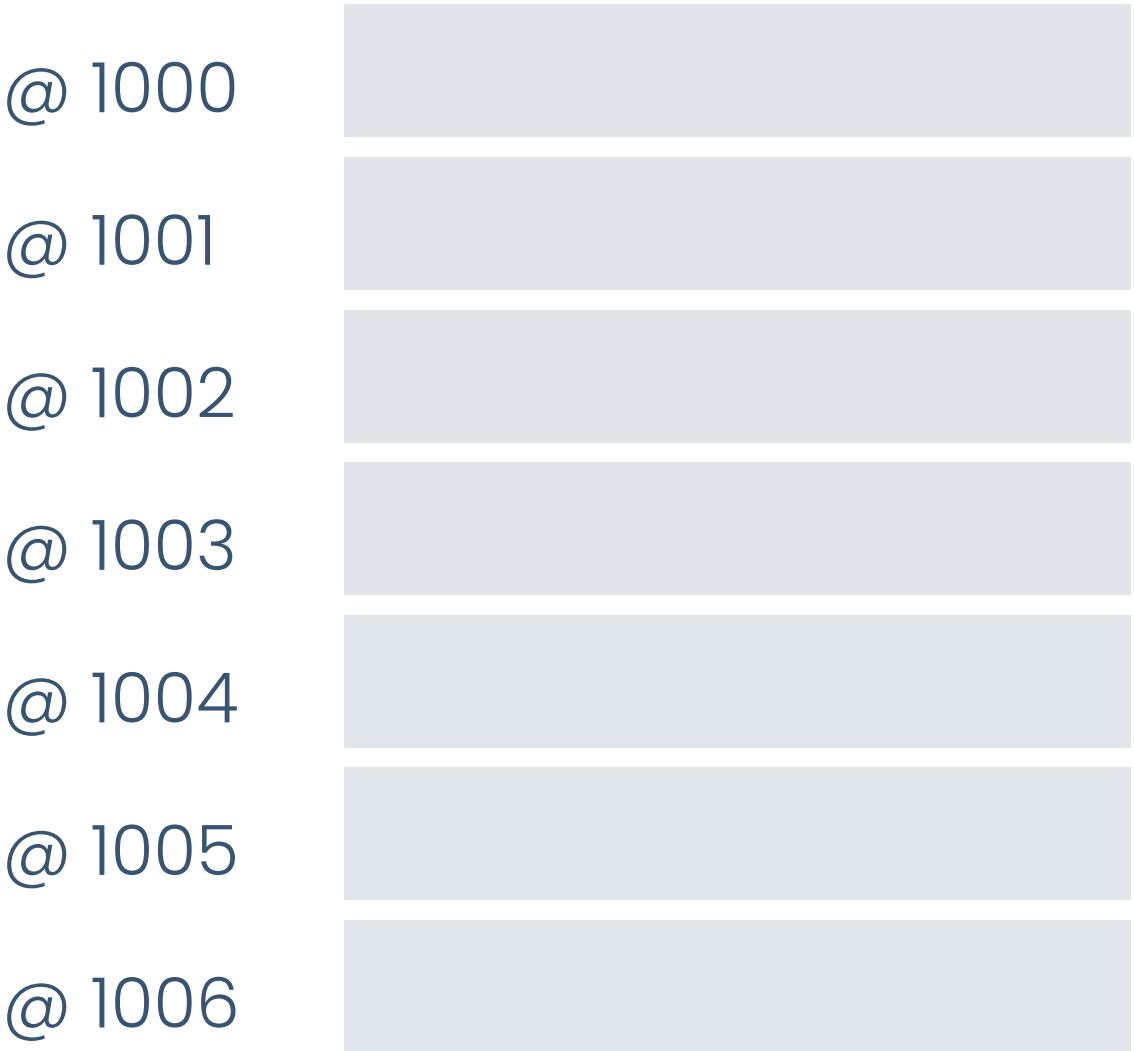
Escriu un algorisme que:

- Declari una variable entera que es digui “e”
- Li assigni el valor 500
- Declari una variable tipus punter que apunti a “e”
- Modifiqui, a través del punter, el valor de “e”. Volem sumar-li 5 a “e”

```
algorisme exercici és
var

fvar
inici

falgorisme
```



Punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Exercici per practicar

Escriu un algorisme que:

- Declari una variable entera que es digui "e"
- Li assigni el valor 500
- Declari una variable tipus punter que apunti a "e"
- Modifiqui, a través del punter, el valor de "e". Volem sumar-li 5 a "e"

algorisme exercici **és**

var

e : enter;

fvar

inici

falgorisme

@ 1000

e @ 1001

@ 1002

@ 1003

@ 1004

@ 1005

@ 1006

Punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Exercici per practicar

Escriu un algorisme que:

- Declari una variable entera que es digui "e"
- Li assigni el valor 500
- Declari una variable tipus punter que apunti a "e"
- Modifiqui, a través del punter, el valor de "e". Volem sumar-li 5 a "e"

algorisme exercici **és**

var

e : enter;

p : punter_a_enter;

fvar

inici

falgorisme

@ 1000

e @ 1001

@ 1002

@ 1003

@ 1004

p @ 1005

@ 1006

Punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Exercici per practicar

Escriu un algorisme que:

- Declari una variable entera que es digui "e"
- Li assigni el valor 500
- Declari una variable tipus punter que apunti a "e"
- Modifiqui, a través del punter, el valor de "e". Volem sumar-li 5 a "e"

algorisme exercici és

var

e : enter;

p : punter_a_enter;

fvar

inici

e := 500;

falgorisme

@ 1000

e @ 1001

500

@ 1002

@ 1003

@ 1004

p @ 1005

@ 1006

Punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Exercici per practicar

Escriu un algorisme que:

- Declari una variable entera que es digui "e"
- Li assigni el valor 500
- Declari una variable tipus punter que apunti a "e"
- Modifiqui, a través del punter, el valor de "e". Volem sumar-li 5 a "e"

algorisme exercici **és**

var

 e : enter;

 p : punter_a_enter;

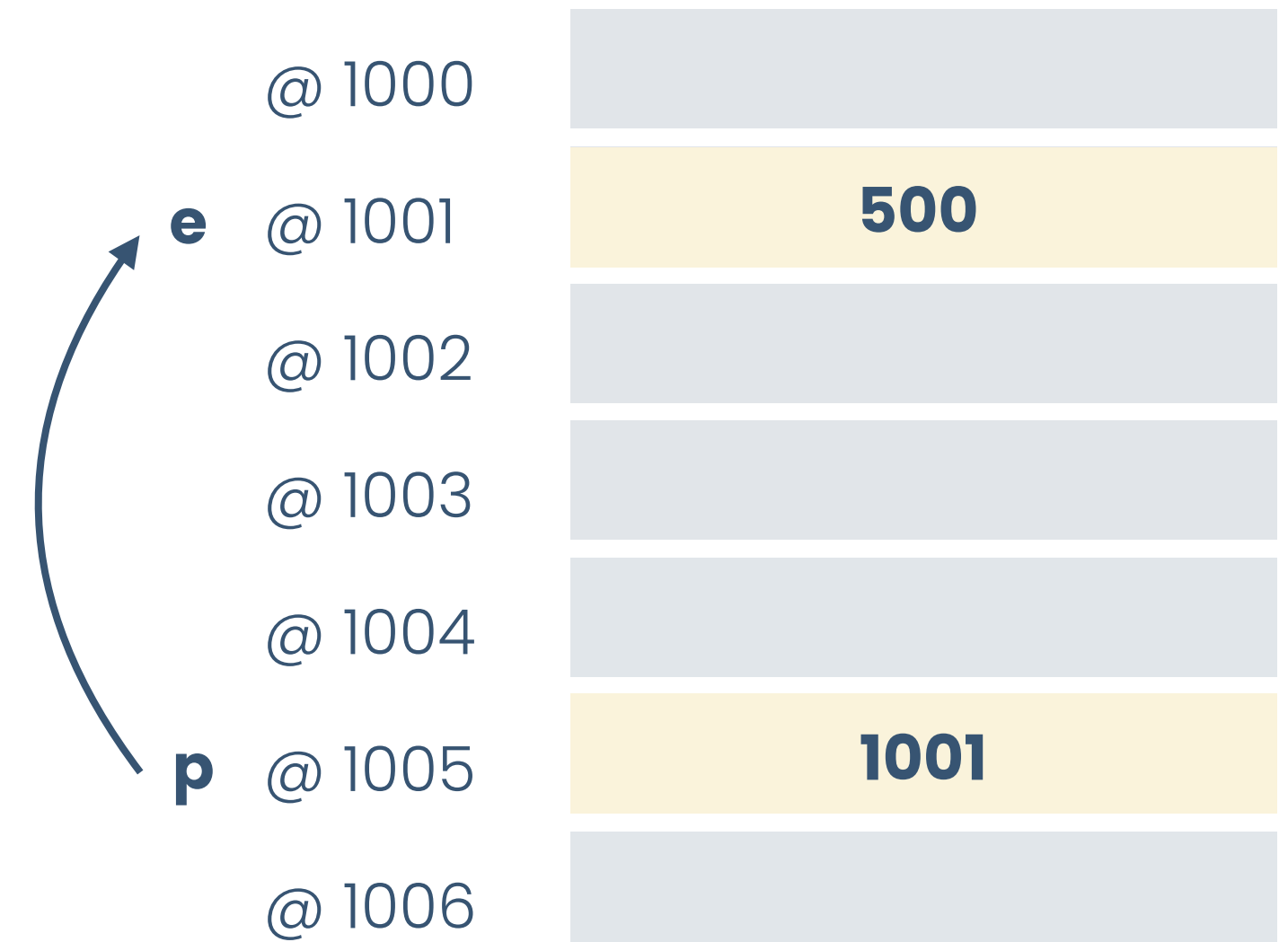
fvar

inici

 e := 500;

 p := &e; // p apunta a e

falgorisme



Punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Exercici per practicar

Escriu un algorisme que:

- Declari una variable entera que es digui "e"
- Li assigni el valor 500
- Declari una variable tipus punter que apunti a "e"
- Modifiqui, a través del punter, el valor de "e". Volem sumar-li 5 a "e"

algorisme exercici **és**

var

 e : enter;

 p : punter_a_enter;

fvar

inici

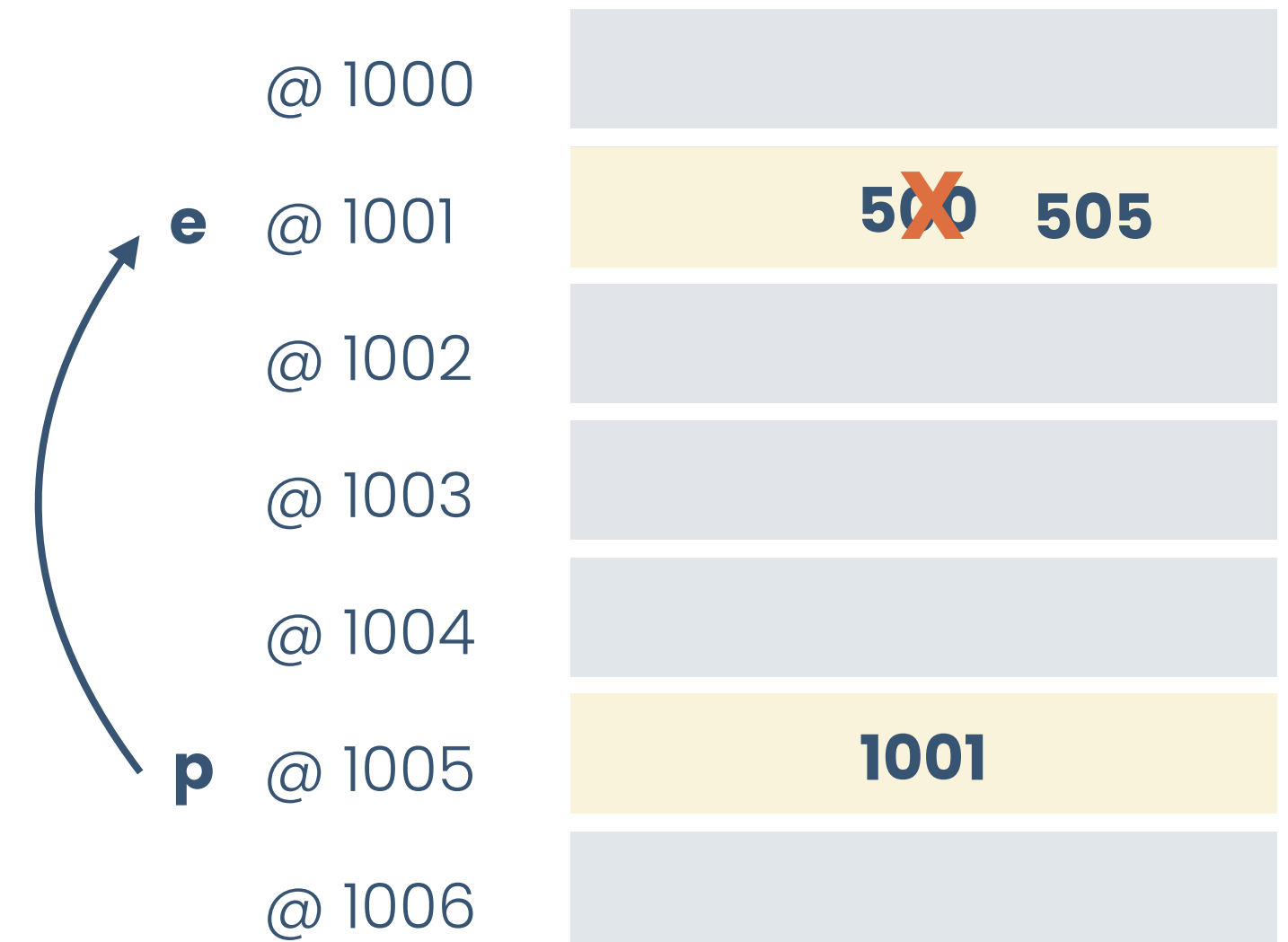
 e := 500;

 p := &e; // p apunta a e

 *p := *p + 5; // e ara

 valdrà 505

falgorisme



Punters

Operador "&": et dóna l'adreça de memòria d'un objecte
Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Exercici per practicar II

Escriu un algorisme que declari dues variables enteres **a** i **b** i en faci un intercanvi de valors fent servir punters.

Pistes: necessitaràs dos punters i una variable auxiliar

algorisme intercanvi és
var

fvar
inici

falgorisme

@ 1500	
@ 1501	
@ 1502	
@ 1503	
@ 1504	
@ 1505	
@ 1506	
@ 1507	
@ 1508	
@ 1509	

Punters

Operador "&": et dóna l'adreça de memòria d'un objecte
Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

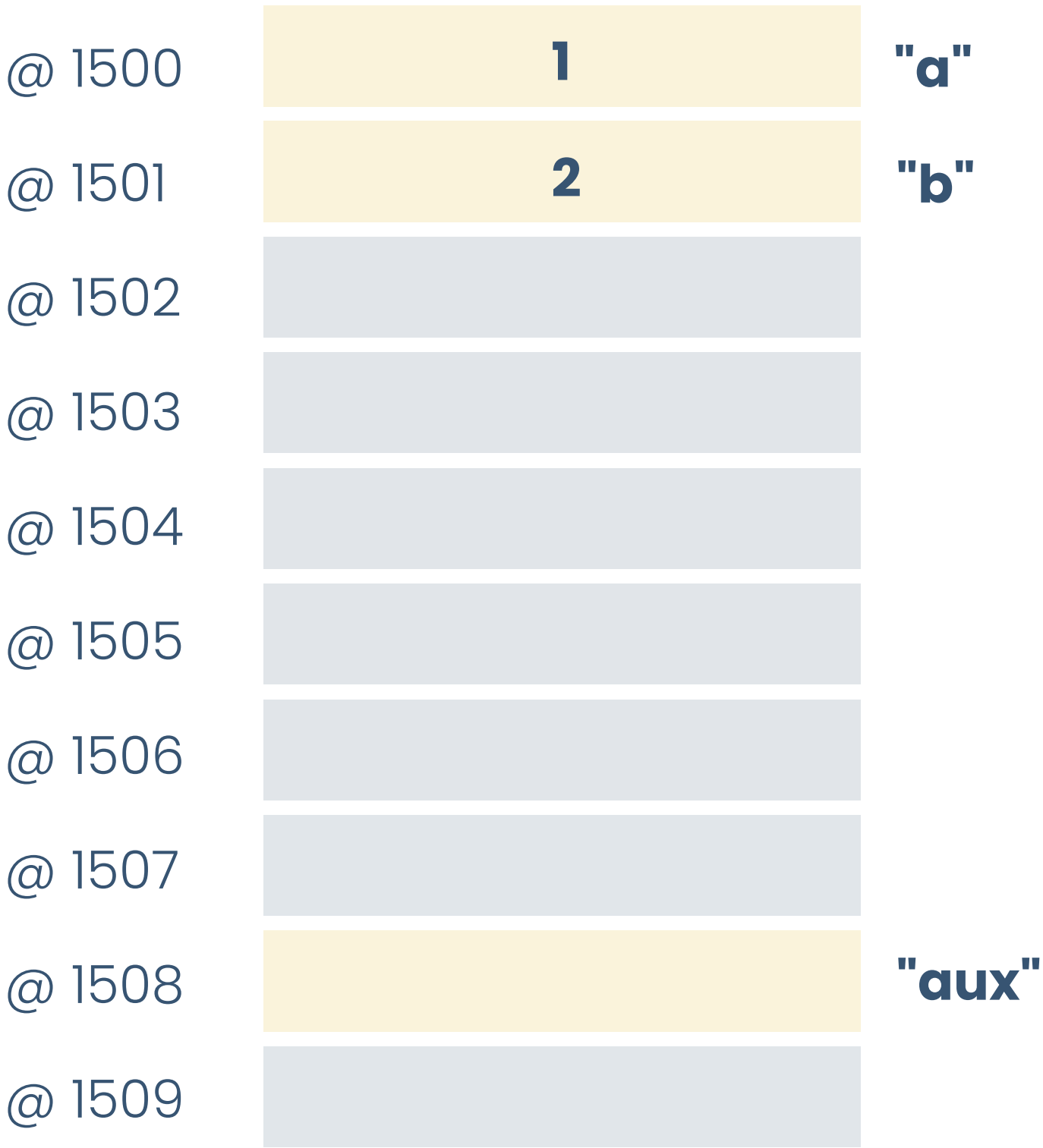
Exercici per practicar II

Escriu un algorisme que declari dues variables enteres **a** i **b** i en faci un intercanvi de valors fent servir punters.

Pistes: necessitaràs dos punters i una variable auxiliar

```
algorisme intercanvi és
var
  a, b: enter;
  aux: enter;
  pa: punter_a_enter;
  pb: punter_a_enter;
fvar
inici
  a := 1;
  b := 2;

falgorisme
```



Punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Exercici per practicar II

Escriu un algorisme que declari dues variables enteres **a** i **b** i en faci un intercanvi de valors fent servir punters.

Pistes: necessitaràs dos punters i una variable auxiliar

algorisme intercanvi **és**
var

a, b: enter;

aux: enter;

pa: punter_a_enter;

pb: punter_a_enter;

fvar

inici

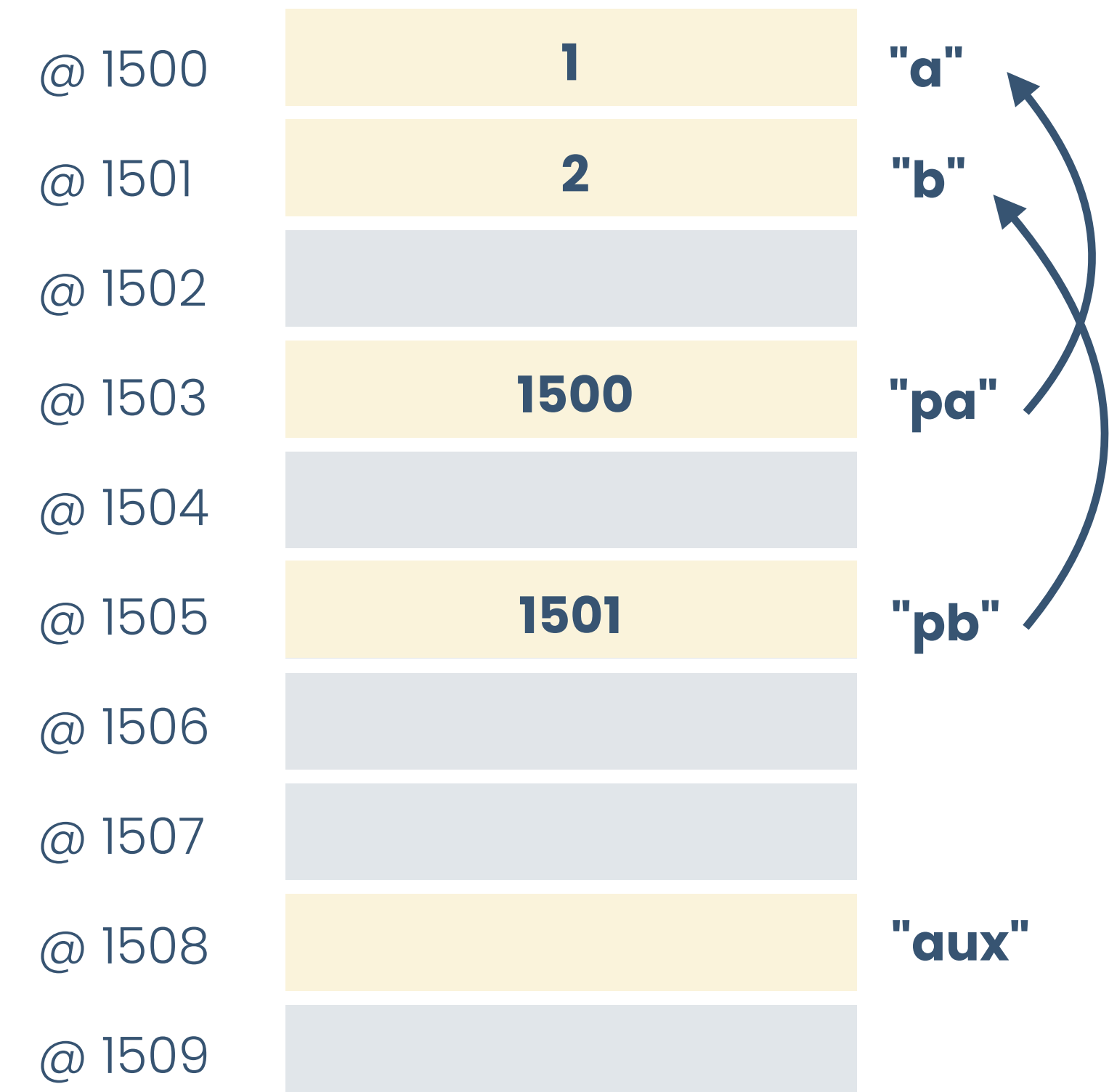
a := 1;

b := 2;

pa := &a;

pb := &b;

falgorisme



Punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Exercici per practicar II

Escriu un algorisme que declari dues variables enteres **a** i **b** i en faci un intercanvi de valors fent servir punters.

Pistes: necessitaràs dos punters i una variable auxiliar

algorisme intercanvi **és**

var

a, b: enter;

aux: enter;

pa: punter_a_enter;

pb: punter_a_enter;

fvar

inici

a := 1;

b := 2;

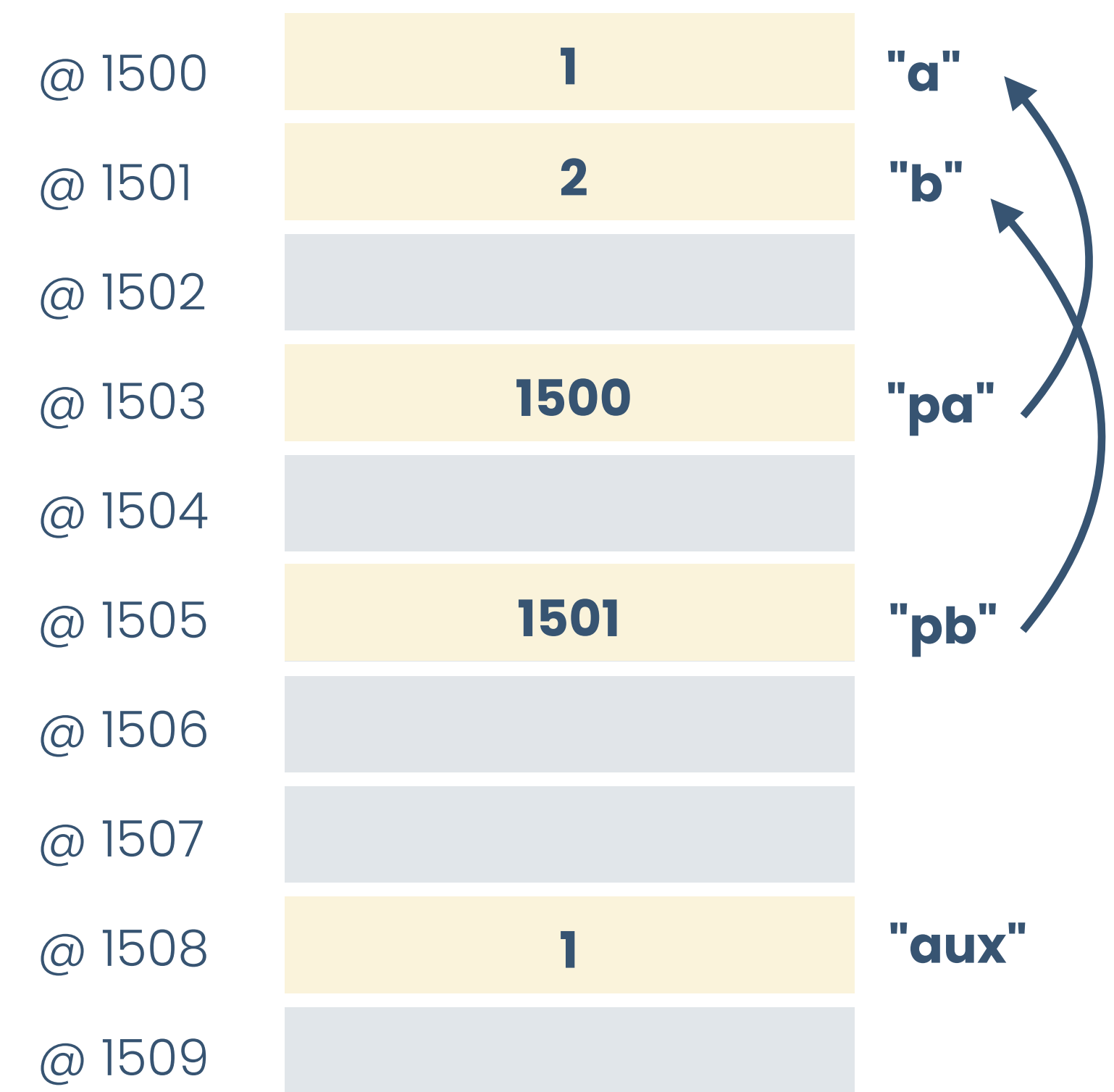
pa := &a;

pb := &b;

aux := *pa;

falgorisme

Guardem en l'enter **aux** allò on apunta **pa** (o sigui, **a**) (aux val 1)



Punters

Operador "&": et dóna l'adreça de memòria d'un objecte
Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Exercici per practicar II

Escriu un algorisme que declari dues variables enteres **a** i **b** i en faci un intercanvi de valors fent servir punters.

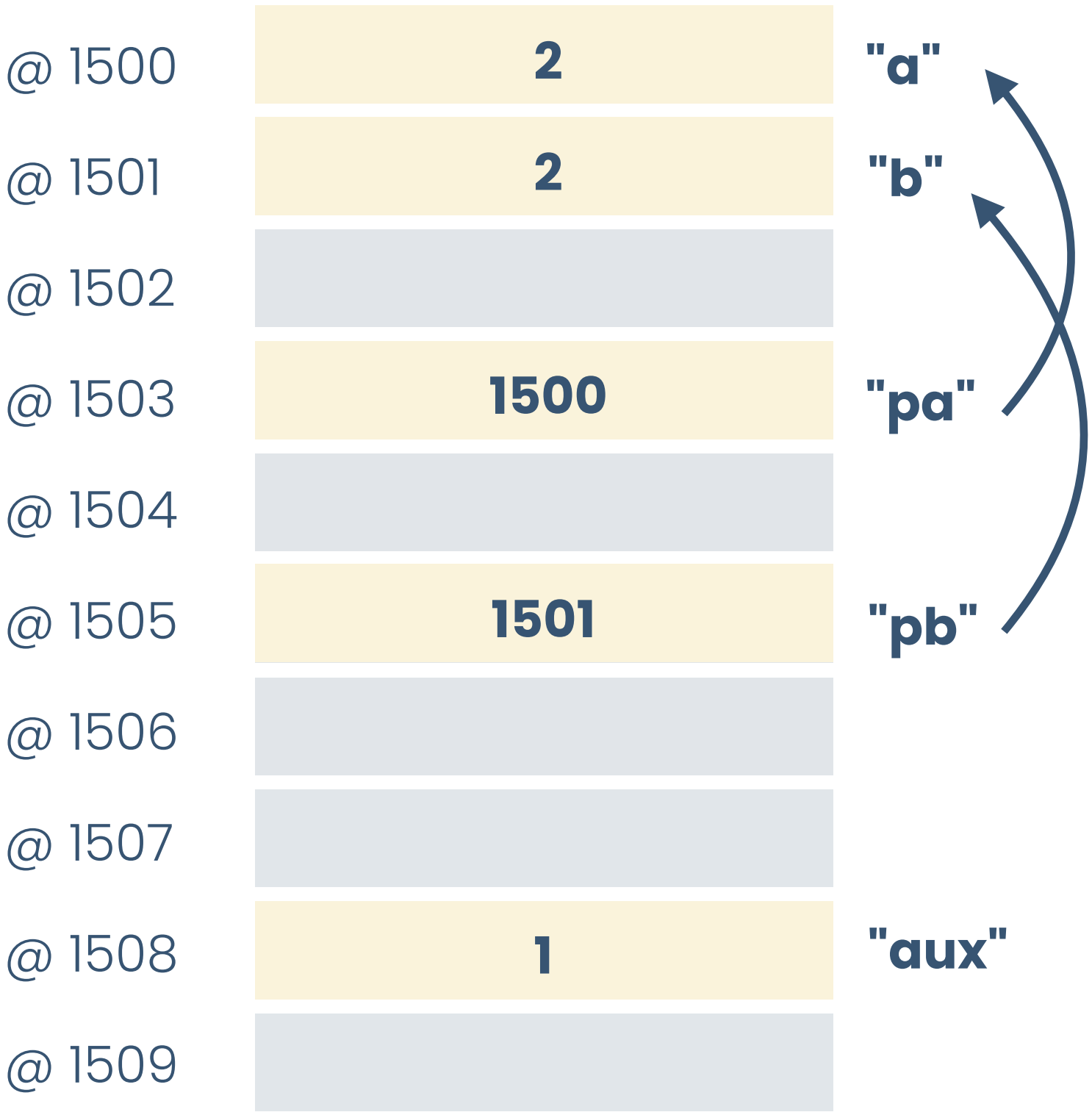
Pistes: necessitaràs dos punters i una variable auxiliar

```
algorisme intercanvi és
var
  a, b: enter;
  aux: enter;
  pa: punter_a_enter;
  pb: punter_a_enter;
fvar
inici
  a := 1;
  b := 2;
  pa := &a;
  pb := &b;
  aux := *pa;
  *pa := *pb;

falgorisme
```

Guardem en l'enter **aux** allò on apunta **pa** (o sigui, **a**) (aux val 1)

Allò on apunta **pa** (o sigui, **a**) ara val allò on apunta **pb** (o sigui, **b**)



Punters

Operador "&": et dóna l'adreça de memòria d'un objecte
Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Exercici per practicar II

Escriu un algorisme que declari dues variables enteres **a** i **b** i en faci un intercanvi de valors fent servir punters.

Pistes: necessitaràs dos punters i una variable auxiliar

algorisme intercanvi és
var

 a, b: enter;
 aux: enter;
 pa: punter_a_enter;
 pb: punter_a_enter;

fvar

inici

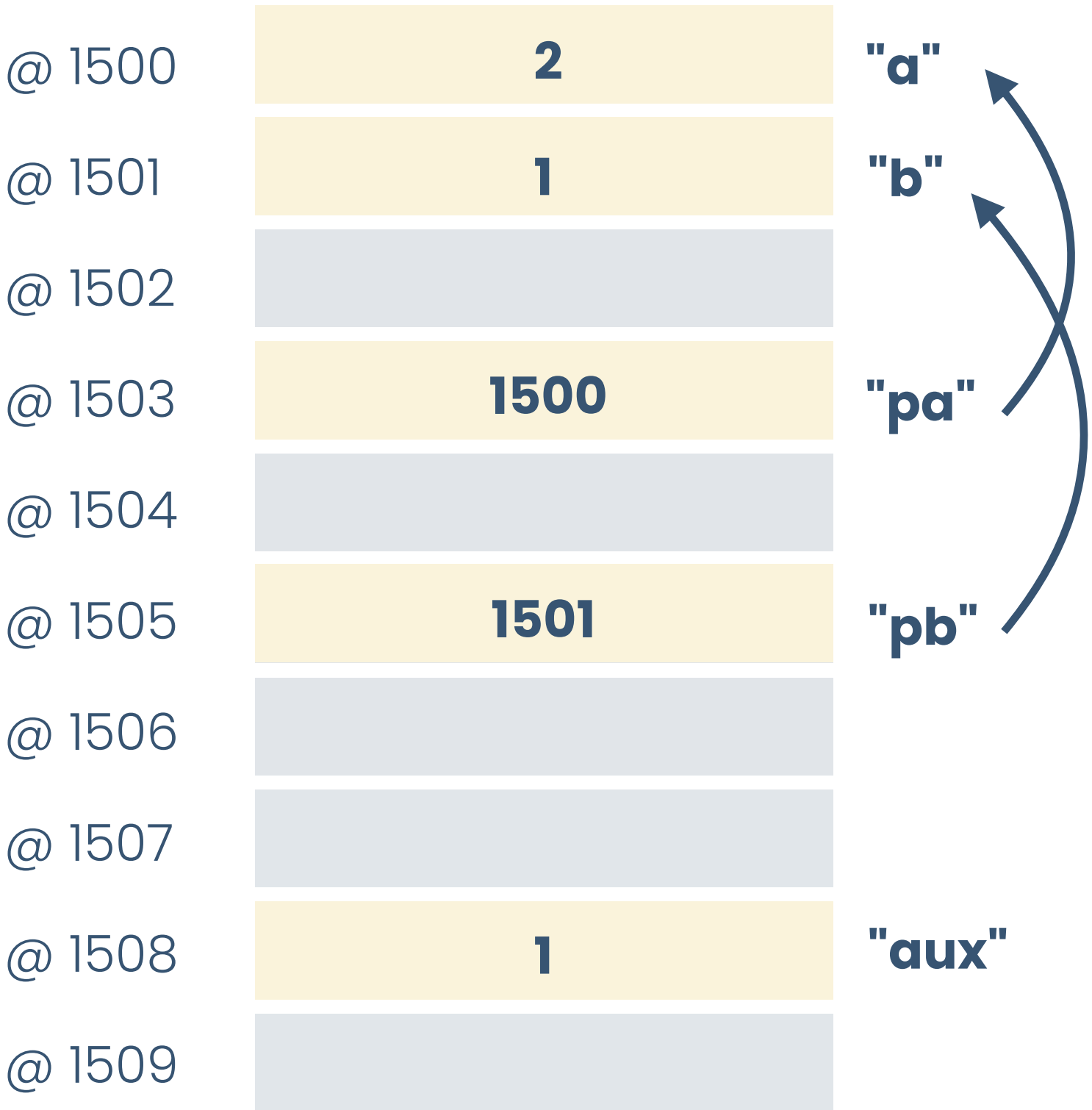
 a := 1;
 b := 2;
 pa := &a;
 pb := &b;
 aux := *pa;
 *pa := *pb;
 *pb := aux;

falgorisme

Guardem en l'enter **aux** allò on apunta **pa** (o sigui, **a**) (aux val 1)

Allò on apunta **pa** (o sigui, **a**)
ara val allò on apunta **pb** (o sigui, **b**)

Allò on apunta **pb** (o sigui, **b**)
ara val el valor que guardava **aux** (o sigui, 1)



Un error a evitar

Aplicar l'operador de desreferència a un punter no inicialitzat.

```
algorisme error_desreferencia és  
var  
    x : enter;  
    p : punter_a_enter;  
fvar  
inici  
    x := 10;  
    escriure(*p);  
falgorisme
```

Un error a evitar

Aplicar l'operador de desreferència a un punter no inicialitzat.

```
algorisme error_desreferencia és  
var  
    x : enter;  
    p : punter_a_enter;  
fvar  
inici  
    x := 10;  
    escriure(*p);  
falgorisme
```

Pretenem escriure els continguts d'allà on apunta "p". Però no li hem dit on ha d'apuntar "p". On apunta?

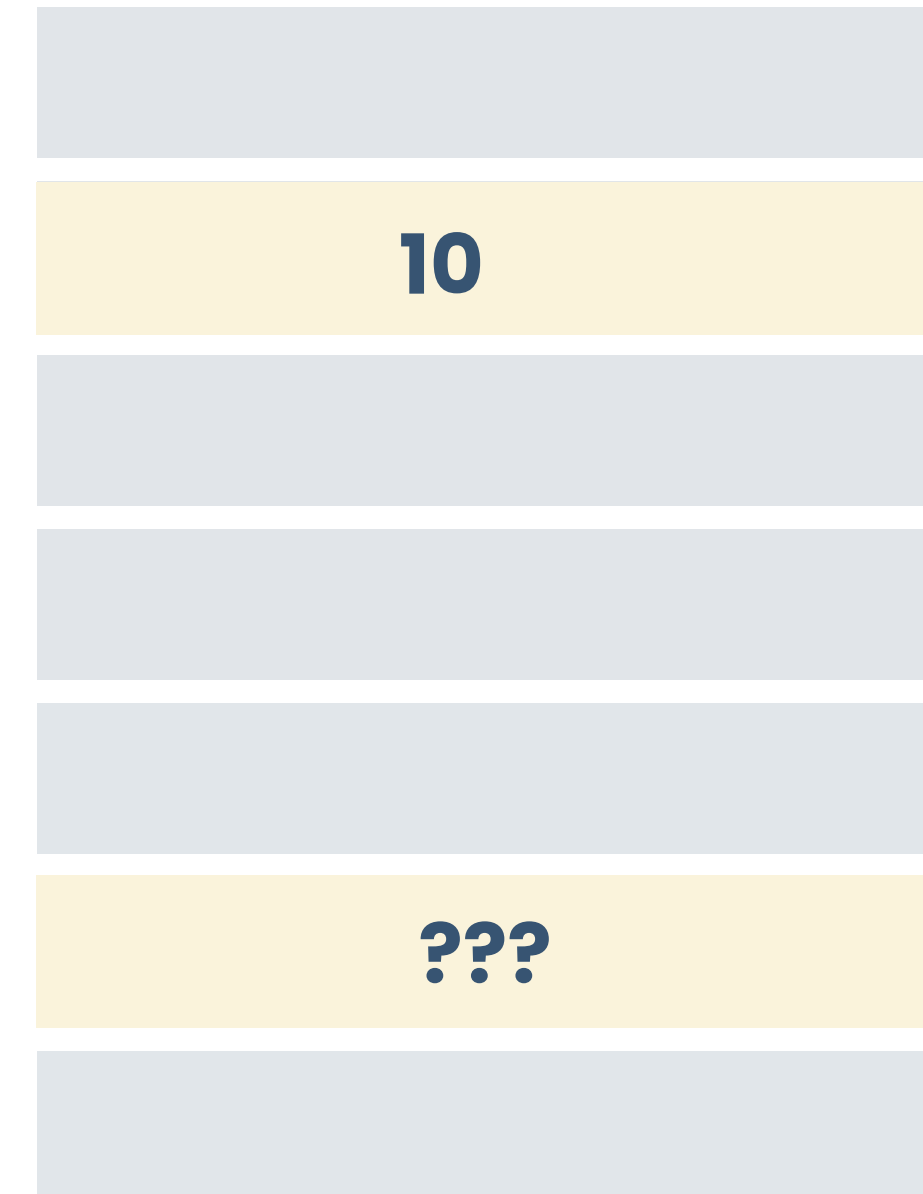
Un error a evitar

Aplicar l'operador de desreferència a un punter no inicialitzat.

```
algorisme error_desreferencia és
var
    x : enter;
    p : punter_a_enter;
fvar
inici
    x := 10;
    escriure(*p);
falgorisme
```

Pretenem escriure els continguts d'allà on apunta "p". Però no li hem dit on ha d'apuntar "p". On apunta?

@ 1000
x @ 1001
@ 1002
@ 1003
@ 1004
p @ 1005
@ 1006



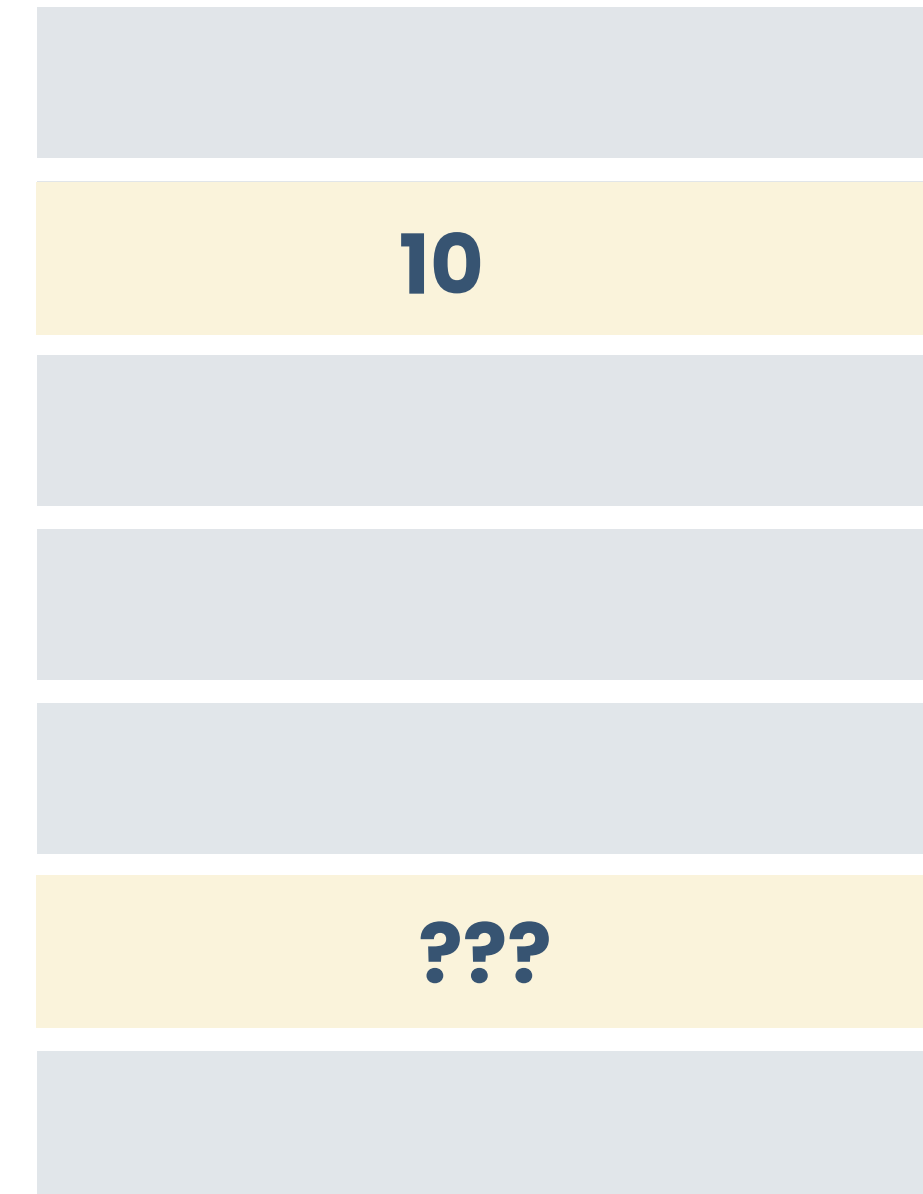
Un error a evitar

Aplicar l'operador de desreferència a un punter no inicialitzat.

```
algorisme error_desreferencia és
var
    x : enter;
    p : punter_a_enter;
fvar
inici
    x := 10;
    escriure(*p);
falgorisme
```

Pretenem escriure els continguts d'allà on apunta "p". Però no li hem dit on ha d'apuntar "p". On apunta?

@ 1000
x @ 1001
@ 1002
@ 1003
@ 1004
p @ 1005
@ 1006



Què passaria?

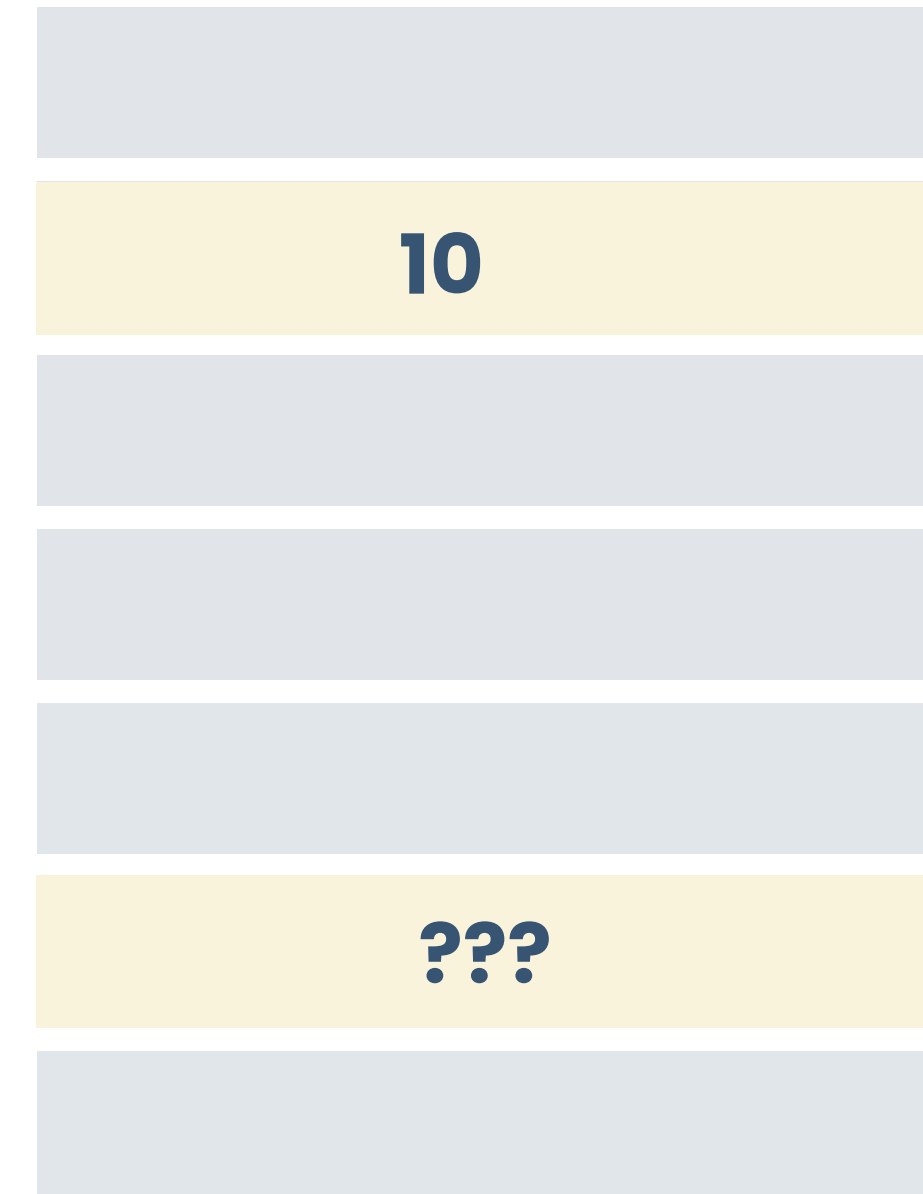
Un error a evitar

Aplicar l'operador de desreferència a un punter no inicialitzat.

```
algorisme error_desreferencia és
var
    x : enter;
    p : punter_a_enter;
fvar
inici
    x := 10;
    escriure(*p);
falgorisme
```

Pretenem escriure els continguts d'allà on apunta "p". Però no li hem dit on ha d'apuntar "p". On apunta?

@ 1000
x @ 1001
@ 1002
@ 1003
@ 1004
p @ 1005
@ 1006



Què passaria?

- Com que el valor de "p" és brossa, i li apliquem l'operador de desreferència, vol dir que anem a la posició de memòria que indica "p" i n'intentem imprimir els continguts.
- Si a "p" hi ha una adreça de memòria vàlida, imprimirem els continguts d'aquella cel·la de memòria (que no ens interessin)
- Si hi ha una adreça de memòria no vàlida, el comportament és indeterminat.

Un error a evitar

Aplicar l'operador de desreferència a un punter no inicialitzat.

I ara, què passaria?

```
algorisme error_desreferencia és
var
    x : enter;
    p : punter_a_enter;
fvar
inici
    x := 10;
    *p := 1;
falgorisme
```

Pretenem canviar el valor d'allà on
apunta "p". Però on apunta?

	@ 1000
x	@ 1001
	@ 1002
	@ 1003
	@ 1004
p	@ 1005
	@ 1006

10

???

Un error a evitar

Aplicar l'operador de desreferència a un punter no inicialitzat.

I ara, què passaria?

```
algorisme error_desreferencia és
var
    x : enter;
    p : punter_a_enter;
fvar
inici
    x := 10;
    *p := 1;
falgorisme
```

Pretenem canviar el valor d'allà on
apunta "p". Però on apunta?

@ 1000
x @ 1001
@ 1002
@ 1003
@ 1004
p @ 1005
@ 1006

10

???

- El programa interpretarà el valor de "p", encara que sigui brossa, com una adreça de memòria a accedir.
- Si l'adreça és vàlida, canviarà el valor d'aquella casella de memòria! On possiblement hi tinguem alguna cosa guardada! (Un error molt difícil de trobar!)

PAS DE PARÀMETRES

EMULAR EL PAS PER REFERÈNCIA

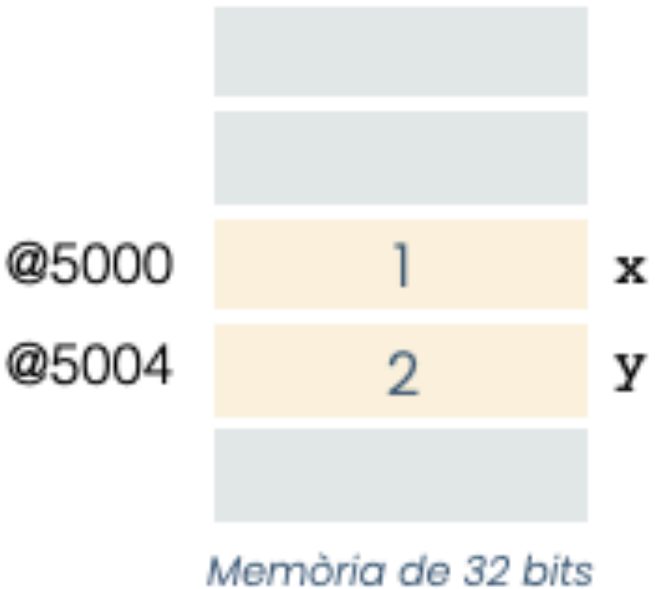
Utilitat dels punters

Pas de paràmetres per referència

Pas de paràmetres

Pas per valor vs. pas per referència

- Distinció **molt important**. En programació, distingim entre dos mecanismes de pas de paràmetres:



Pas per valor

En un llenguatge que fa servir pas per valor, el procediment *intercanvia* rep els valors de les variables *x* i *y*, és a dir:

1
2

```
x := 1;  
y := 2;  
...  
intercanvia  
...  
$ x val 1  
$ y val 2
```

Pas per referència

En un llenguatge que fa servir pas per referència, el procediment *intercanvia* rep les adreces de les variables *x* i *y*, és a dir:

5000
5004

Ho aprendrem a fer quan vegem punters

```
...  
$ x val 2  
$ y val 1  
...  
);
```

Utilitat dels punters

Emulant el pas de paràmetres per referència

Utilitat dels punters

Emulant el pas de paràmetres per referència

- Sense procediments: prog. pral.

```
algorisme intercanvi és  
var  
  a, b, aux: enter;  
fvar  
inici  
  a := 1;  
  b := 2;  
  ...  
  aux := a;  
  a := b;  
  b := aux;  
falgorisme
```

- Quant valen a i b ?

Utilitat dels punters

Emulant el pas de paràmetres per referència

- Sense procediments: prog. pral.
- Amb procediments (pas per valor)

```
algorisme intercanvi és  
var  
  a, b, aux: enter;  
fvar  
inici  
  a := 1;  
  b := 2;  
  ...  
  aux := a;  
  a := b;  
  b := aux;  
falgorisme
```

- Quant valen a i b ?

```
acció intercanvi (a: enter,  
b: enter) és  
var  
  aux: enter;  
fvar  
inici  
  aux := a;  
  a := b;  
  b := aux;  
facció
```

```
...  
a := 1;  
b := 2;  
intercanvi(a,b);  
...
```

- Quant valen a i b ?

Utilitat dels punters

Emulant el pas de paràmetres per referència

- Sense procediments: prog. pral.

```
algorisme intercanvi és  
var  
    a, b, aux: enter;  
fvar  
inici  
    a := 1;  
    b := 2;  
    ...  
    aux := a;  
    a := b;  
    b := aux;  
falgorisme
```

- Quant valen a i b ?

- Amb procediments (pas per valor)

```
acció intercanvi (a: enter,  
b: enter) és  
var  
    aux: enter;  
fvar  
inici  
    aux := a;  
    a := b;  
    b := aux;  
facció
```

```
...  
a := 1;  
b := 2;  
intercanvi(a,b);  
...
```

- Quant valen a i b ?

- Amb procediments (pas per referència)

```
acció intercanvi (pa:  
punter_a_enter, pb:  
punter_a_enter) és  
var  
    aux: enter;  
fvar  
inici  
    aux := *pa;  
    *pa := *pb;  
    *pb := aux;  
facció
```

```
...  
a := 1;  
b := 2;  
intercanvi(&a,&b);  
...
```

- Quant valen a i b ?

Utilitat dels punters

Operador "&": et dóna l'adreça de memòria d'un objecte

Operador "*": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

Emulant el pas de paràmetres per referència

- Com encapsular l'algorisme d'intercanvi dins un procediment?

```
acció intercanvi (pa: punter_a_enter, ←  
pb: punter_a_enter) és  
var  
    aux: enter;  
fvar  
inici  
    aux := *pa;  
    *pa := *pb;  
    *pb := aux;  
facció
```

Quan declaro la meva funció, els paràmetres d'entrada que jo vull modificar seran PUNTERS a aquell tipus.

```
...  
a := 1;  
b := 2;  
intercanvi(&a, &b); ←  
escriure("Després, a val", a, "i b val", b);
```

Quan faig la crida, li passo per paràmetre LES ADRECES DE MEMÒRIA de **a** i **b**

Utilitat dels punters

Exercici de pas de paràmetres per referència

- Escriure, en pseudocodi:
 - Una acció que rebi per 3 paràmetres numèrics **a**, **b** i **c**, i guardi a **c** la suma de **a+b**.
 - El programa principal que crida aquest codi.
- Primer pas: pensar de quin tipus seran les dades, tant al programa principal com a la funció, i escriure la capçalera de la funció

Utilitat dels punters

Exercici de pas de paràmetres per referència

- Escriure, en pseudocodi:
 - Una acció que rebi per 3 paràmetres numèrics **a**, **b** i **c**, i guardi a **c** la suma de **a+b**.
 - El programa principal que crida aquest codi.
- Primer pas: pensar de quin tipus seran les dades, tant al programa principal com a la funció, i escriure la capçalera de la funció

```
acció suma (a: enter, b: enter, c: punter_a_enter) és
```

```
inici
```

```
facció
```

Utilitat dels punters

Exercici de pas de paràmetres per referència

- Escriure, en pseudocodi:
 - Una acció que rebi per 3 paràmetres numèrics **a**, **b** i **c**, i guardi a **c** la suma de **a+b**.
 - El programa principal que crida aquest codi.
- Primer pas: pensar de quin tipus seran les dades, tant al programa principal com a la funció, i escriure la capçalera de la funció

```
acció suma (a: enter, b: enter, c: punter_a_enter) és
```

```
inici
```

```
    *c := a + b;
```

```
facció
```

Utilitat dels punters

Exercici de pas de paràmetres per referència

- Escriure, en pseudocodi:
 - Una acció que rebi per 3 paràmetres numèrics **a**, **b** i **c**, i guardi a **c** la suma de **a+b**.
 - El programa principal que crida aquest codi.
- Primer pas: pensar de quin tipus seran les dades, tant al programa principal com a la funció, i escriure la capçalera de la funció

```
acció suma (a: enter, b: enter, c: punter_a_enter) és
```

```
inici
```

```
    *c := a + b;
```

```
facció
```

```
algorisme principal és  
var
```

```
    fvar  
    inici
```

```
    falgorisme
```

Utilitat dels punters

Exercici de pas de paràmetres per referència

- Escriure, en pseudocodi:
 - Una acció que rebi per 3 paràmetres numèrics **a**, **b** i **c**, i guardi a **c** la suma de **a+b**.
 - El programa principal que crida aquest codi.
- Primer pas: pensar de quin tipus seran les dades, tant al programa principal com a la funció, i escriure la capçalera de la funció

```
acció suma (a: enter, b: enter, c: punter_a_enter) és
```

```
inici
```

```
    *c := a + b;
```

```
facció
```

```
algorisme principal és
```

```
var
```

```
    a : enter;
```

```
    b : enter;
```

```
    c : enter;
```

```
fvar
```

```
inici
```

```
    a := 1;
```

```
    b := 2;
```

```
    suma(a,b,&c);
```

```
    escriure("a val", a);
```

```
    escriure("b val", b);
```

```
    escriure("c val", c);
```

```
falgorisme
```


Utilitat dels punters

Exercici de pas de paràmetres per referència (II)

- Gràcies al pas de paràmetres per referència, podem “saltar-nos” la restricció d’haver de retornar només un valor.
- Escriu una acció que, donada una taula de caràcters acabada en ‘\0’, compti el nombre de majúscules, minúscules i dígit.
- Escriu el programa principal que crida aquest codi.

Utilitat dels punters

Exercici de pas de paràmetres per referència (II)

- Gràcies al pas de paràmetres per referència, podem “saltar-nos” la restricció d’haver de retornar només un valor.
- Escriu una acció que, donada una taula de caràcters acabada en ‘\0’, compti el nombre de majúscules, minúscules i dígit.
- Escriu el programa principal que crida aquest codi.

```
acció comptar (v: taula[] de caràcter, n_maj: punter_a_enter,  
n_min: punter_a_enter, n_dig: punter_a_enter) és  
var i: enter; fvar  
inici
```

facció

```
algorisme principal és  
const fconst  
var
```

```
fvar  
inici
```

falgorisme

Utilitat dels punters

Exercici de pas de paràmetres per referència (II)

- Gràcies al pas de paràmetres per referència, podem “saltar-nos” la restricció d’haver de retornar només un valor.
- Escriu una acció que, donada una taula de caràcters acabada en ‘\0’, compti el nombre de majúscules, minúscules i dígit.
- Escriu el programa principal que crida aquest codi.

```
acció comptar (v: taula[] de caràcter, n_maj: punter_a_enter,
n_min: punter_a_enter, n_dig: punter_a_enter) és
var i: enter; fvar
inici
  i := 0;
  *n_maj := 0; *n_min := 0; *n_dig := 0;
mentre ( v[i] ≠ '\0' ) fer
  si (v[i] ≥ 'A' i v[i] ≤ 'Z') llavors
    *n_maj := *n_maj + 1;
  sino si(v[i] ≥ 'a' i v[i] ≤ 'z') llavors
    *n_min := *n_min + 1;
  sino si(v[i] ≥ '0' i v[i] ≤ '9') llavors
    *n_dig := *n_dig + 1;
  fsi
  i := i + 1;
fmentre
facció
```

```
algorisme principal és
const                fconst
var

fvar
inici

falgorisme
```

Utilitat dels punters

Exercici de pas de paràmetres per referència (II)

- Gràcies al pas de paràmetres per referència, podem “saltar-nos” la restricció d’haver de retornar només un valor.
- Escriu una acció que, donada una taula de caràcters acabada en ‘\0’, compti el nombre de majúscules, minúscules i dígit.
- Escriu el programa principal que crida aquest codi.

```
acció comptar (v: taula[] de caràcter, n_maj: punter_a_enter,
n_min: punter_a_enter, n_dig: punter_a_enter) és
var i: enter; fvar
inici
  i := 0;
  *n_maj := 0; *n_min := 0; *n_dig := 0;
mentre ( v[i] ≠ '\0' ) fer
  si (v[i] >= 'A' i v[i] <= 'Z') llavors
    *n_maj := *n_maj + 1;
  sino si(v[i] >= 'a' i v[i] <= 'z') llavors
    *n_min := *n_min + 1;
  sino si(v[i] >= '0' i v[i] <= '9') llavors
    *n_dig := *n_dig + 1;
  fsi
  i := i + 1;
fmentre
facció
```

```
algorisme principal és
const MAX := 100 fconst
var
  par: taula[MAX] de caràcters;
  maj, min, dig: enter;
fvar
inici
  escriure("Introdueix paraula");
  llegir(par);
  comptar(par, &maj, &min, &dig);
  escriure("Majúscules: ", maj);
  escriure("Minúscules: ", min);
  escriure("Dígits: ", dig);
falgorisme
```

Punters i pas per referència

Exercici: fes el seguiment de les variables a memòria en la crida d'aquesta funció i digues quin valor tenen "i" i "k"

```
acció processa_valors (a: punter_a_enter, b: punter_a_enter) és
```

```
inici
```

```
    *a := *a + 10;
```

```
    *b := *a - *b;
```

```
facció
```

```
algorisme principal és
```

```
var
```

```
    i, k: enter;
```

```
fvar
```

```
inici
```

```
    i := 3;
```

```
    k := 7;
```

```
    processa_valors (&i, &k);
```

```
    escriure("Els valors són i=", i, ", k=", k);
```

```
falgorisme
```

Punters i pas per referència

Fes el mateix amb aquesta variació del codi anterior:

```
acció processa_valors (i: punter_a_enter, k: punter_a_enter) és  
var  
    j: punter_a_enter;  
fvar  
inici  
    j := k;  
    *i := *i + *j;  
    *j := *i - *k;  
    *i := *i - *j;  
facció
```

```
algorisme principal és  
var  
    i, k: enter;  
fvar  
inici  
    i := 3;  
    k := 7;  
    processa_valors (&i, &k);  
    escriure("Els valors són i=", i, ", k=", k);  
falgorisme
```



Dubtes?

Aprofiteu que ara es complica la cosa

DOBLES PUNTERS

Dobles punters

Un punter també pot apuntar a un altre punter !

- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Dobles punters

Un punter també pot apuntar a un altre punter !

- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és  
var
```

```
fvar  
inici
```

```
falgorisme
```

@ 1000

@ 1001

@ 1002

@ 1003

@ 1004

@ 1005

@ 1006

Dobles punters

Un punter també pot apuntar a un altre punter !

- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és
```

```
var
```

```
  v : enter;
```

```
fvar
```

```
inici
```

```
falgorisme
```

@ 1000

@ 1001

@ 1002

@ 1003

@ 1004

@ 1005

@ 1006

Dobles punters

Un punter també pot apuntar a un altre punter !

- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és
```

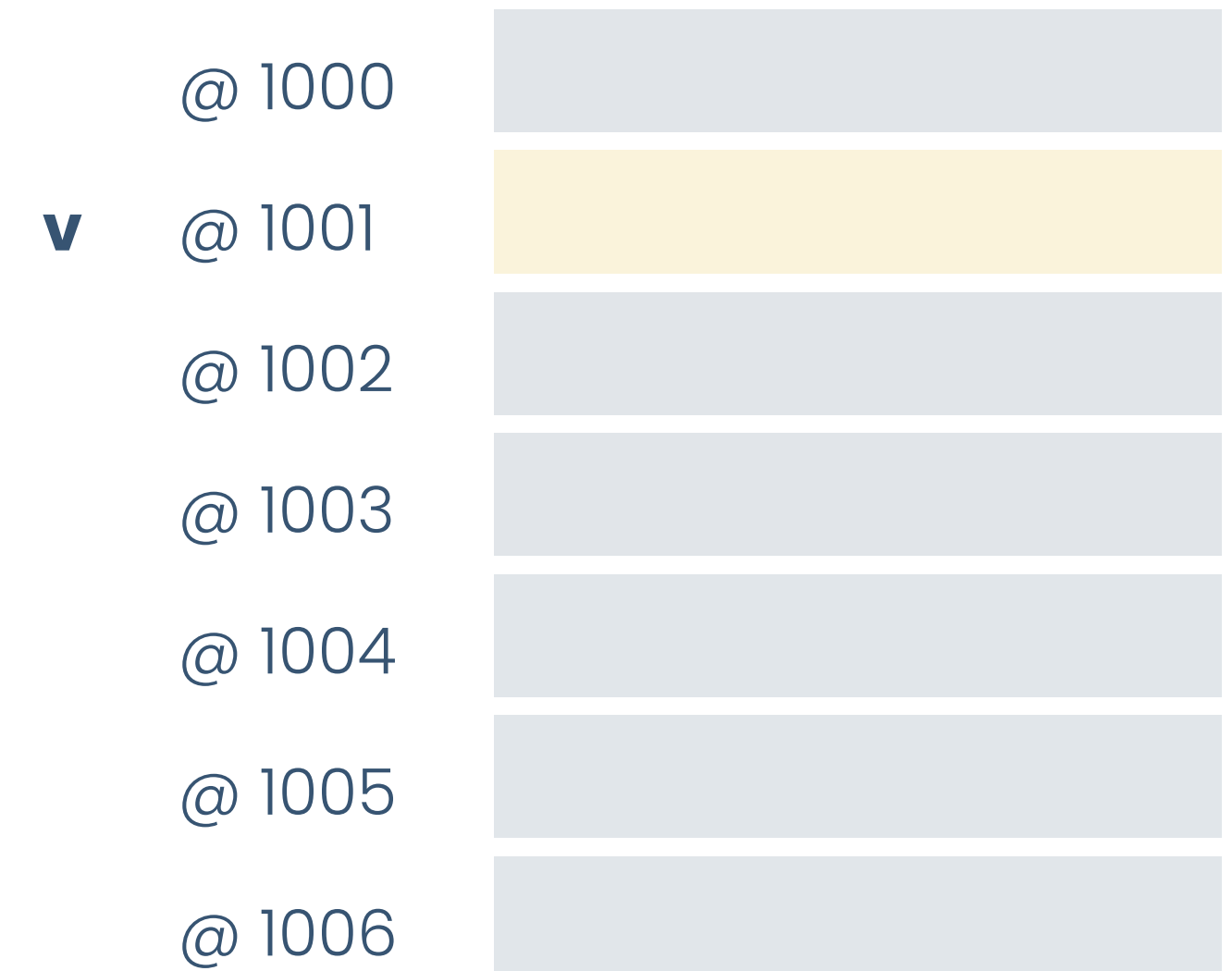
```
var
```

```
  v : enter;
```

```
fvar
```

```
inici
```

```
falgorisme
```



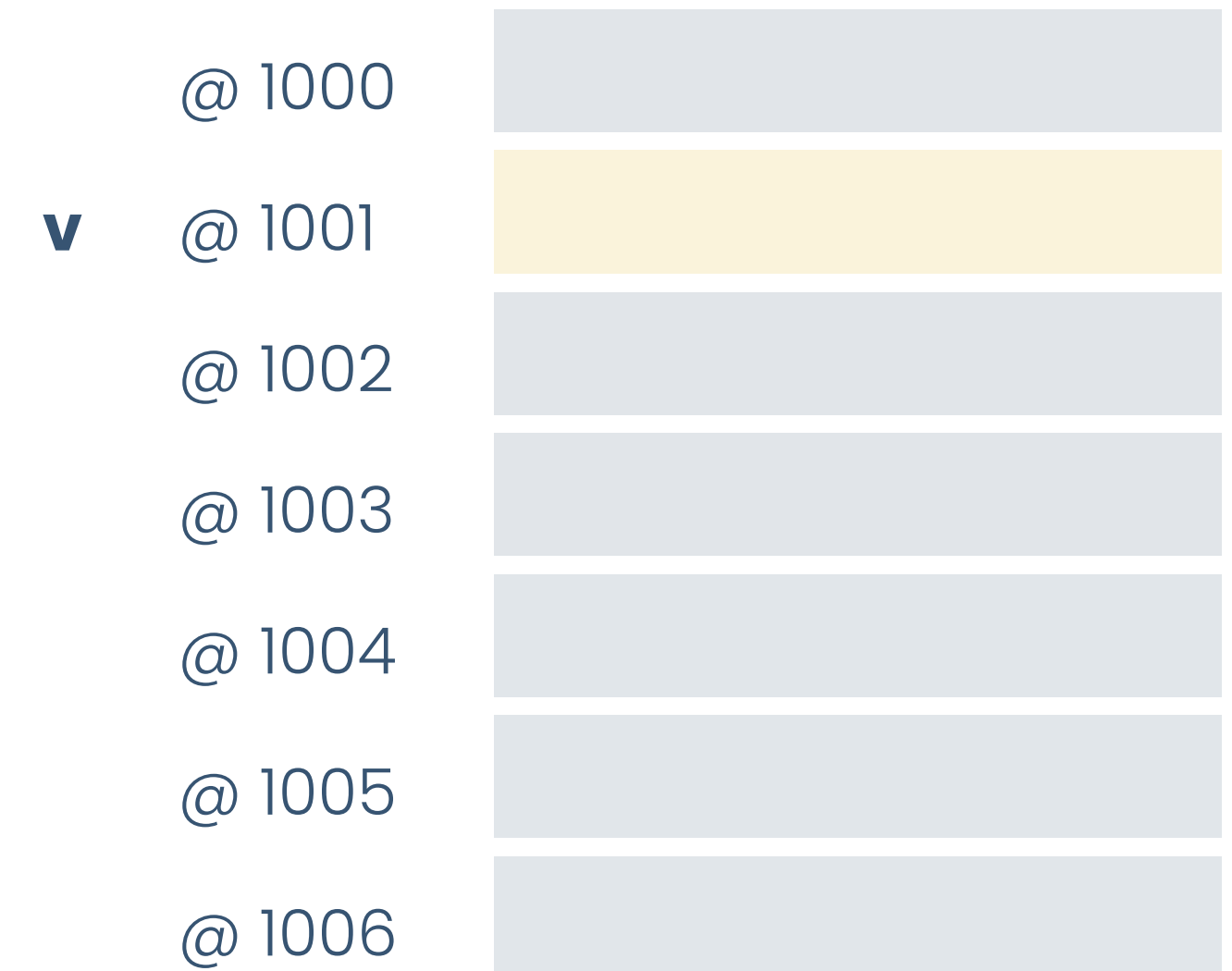
Dobles punters

Un punter també pot apuntar a un altre punter !

- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és  
var  
  v : enter;  
  p : punter_a_enter;  
  
fvar  
inici  
  
  
falgorisme
```



Dobles punters

Un punter també pot apuntar a un altre punter !

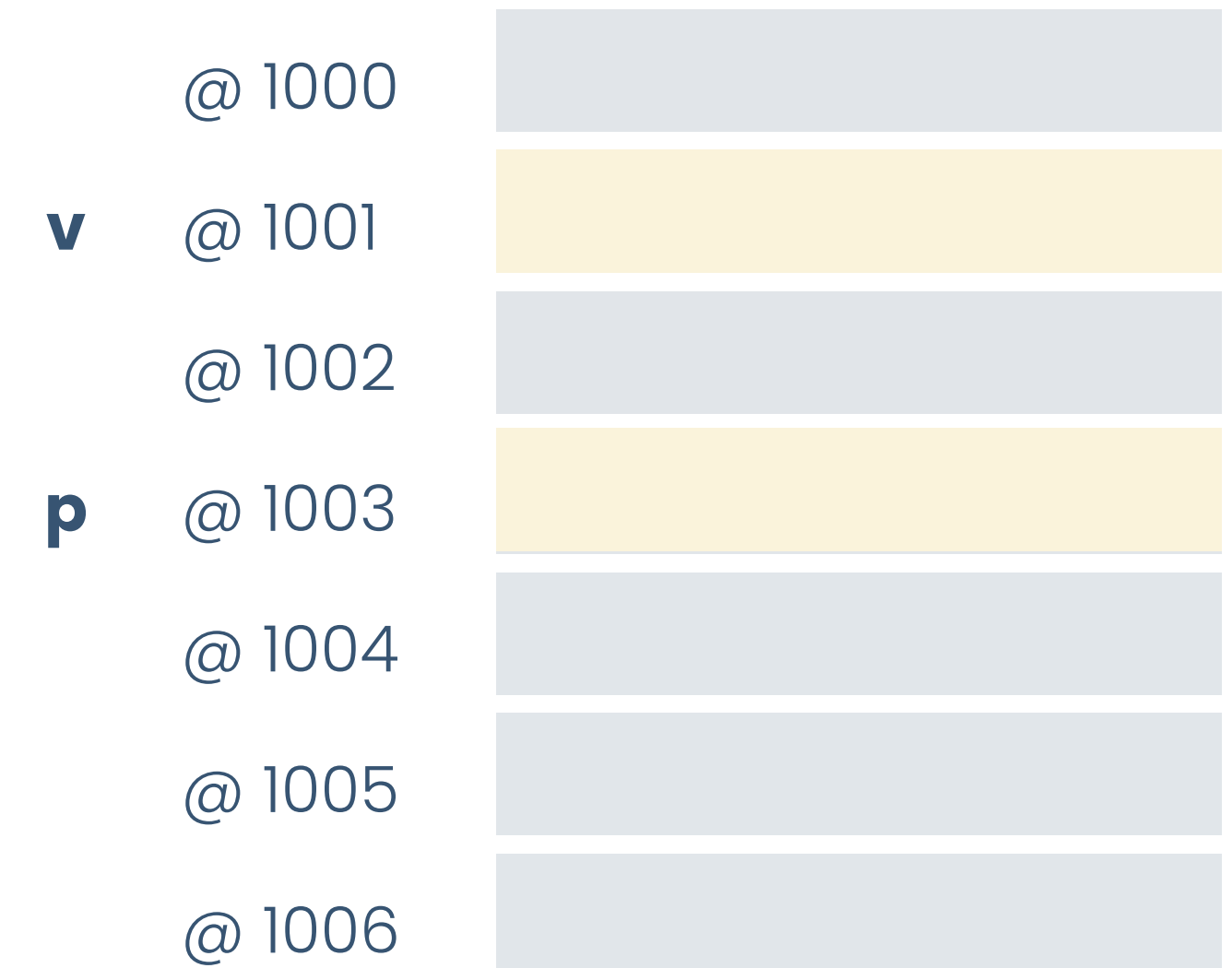
- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;

fvar
inici

falgorisme
```



Dobles punters

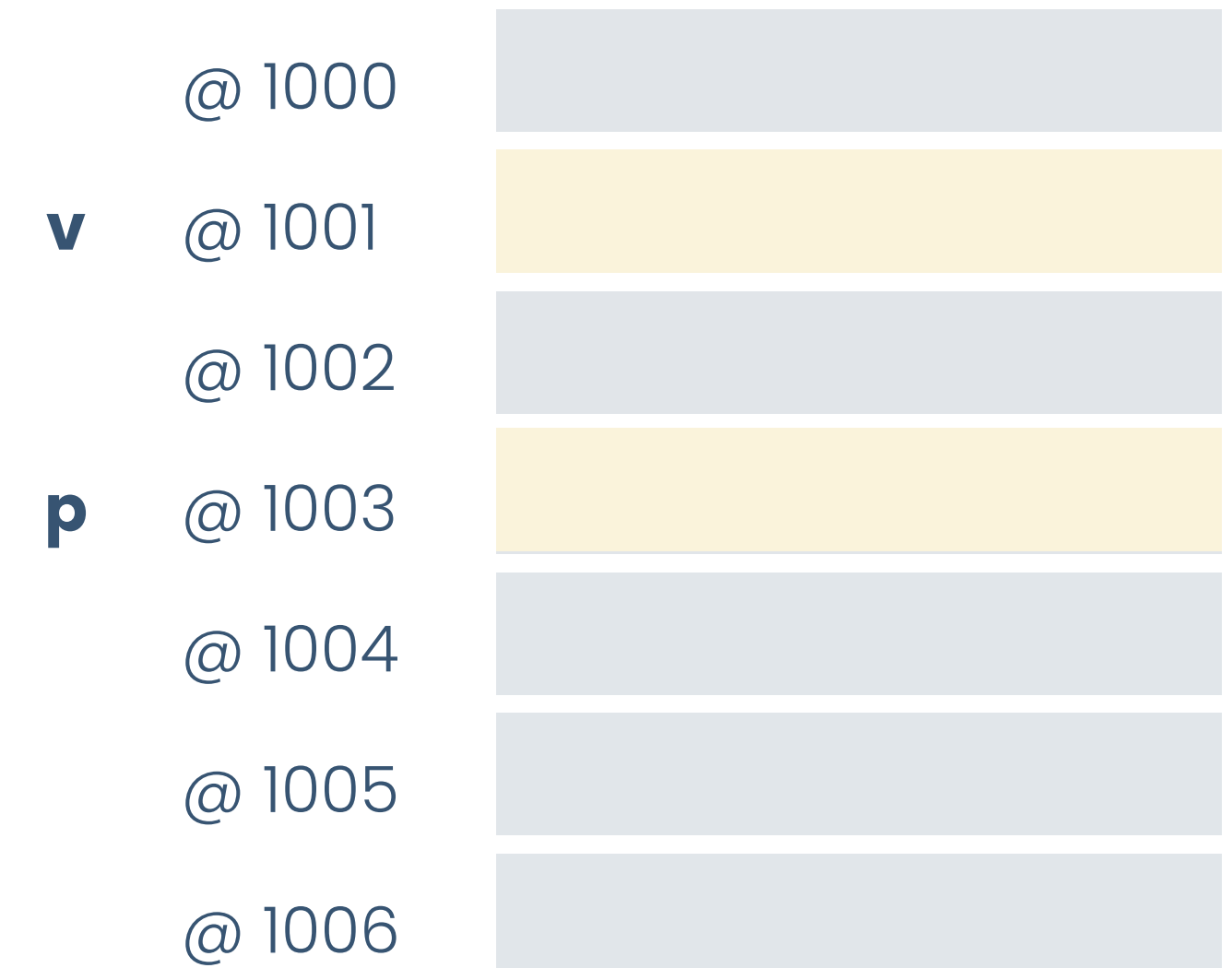
Un punter també pot apuntar a un altre punter !

- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  pp : punter_a_punter_a_enter;
fvar
inici

falgorisme
```



Dobles punters

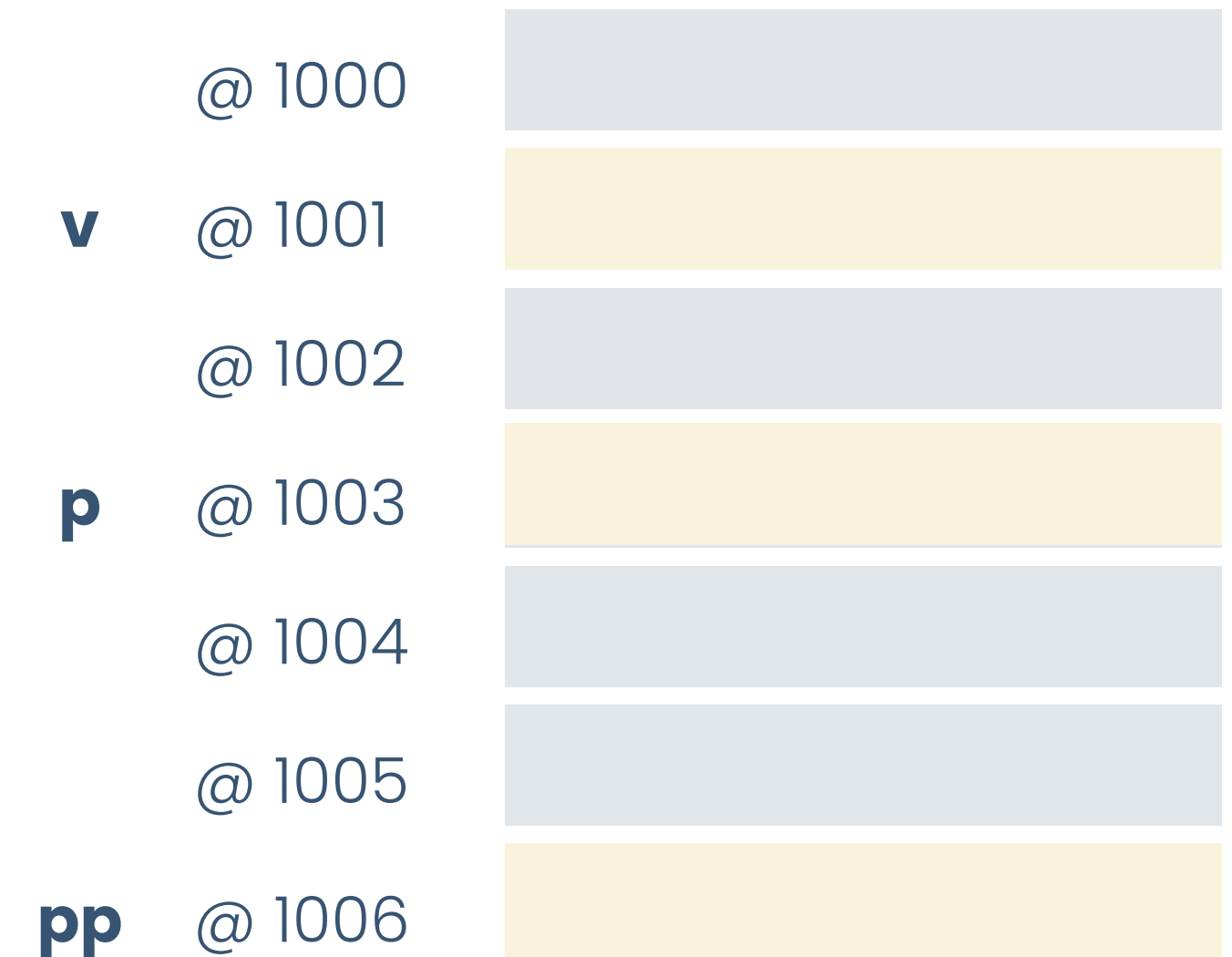
Un punter també pot apuntar a un altre punter !

- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  pp : punter_a_punter_a_enter;
fvar
inici

falgorisme
```



Dobles punters

Un punter també pot apuntar a un altre punter !

- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

algorisme exemple és

var

 v : enter;

 p : punter_a_enter;

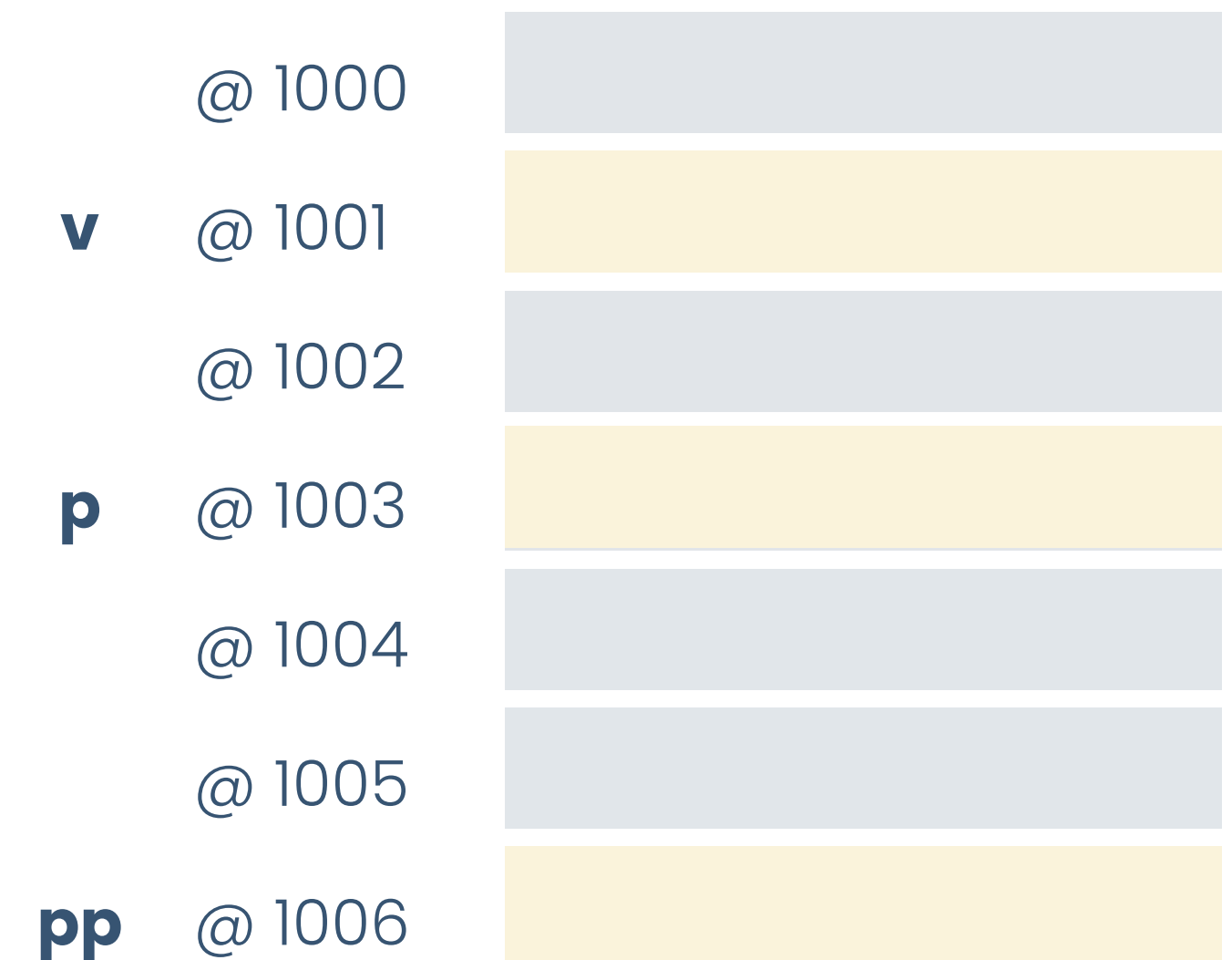
 pp : punter_a_punter_a_enter;

fvar

inici

falgorisme

● → Si us hi fixeu, he de seguir especificant quin és el tipus final a on apunto (aquí, enter).



Dobles punters

Un punter també pot apuntar a un altre punter !

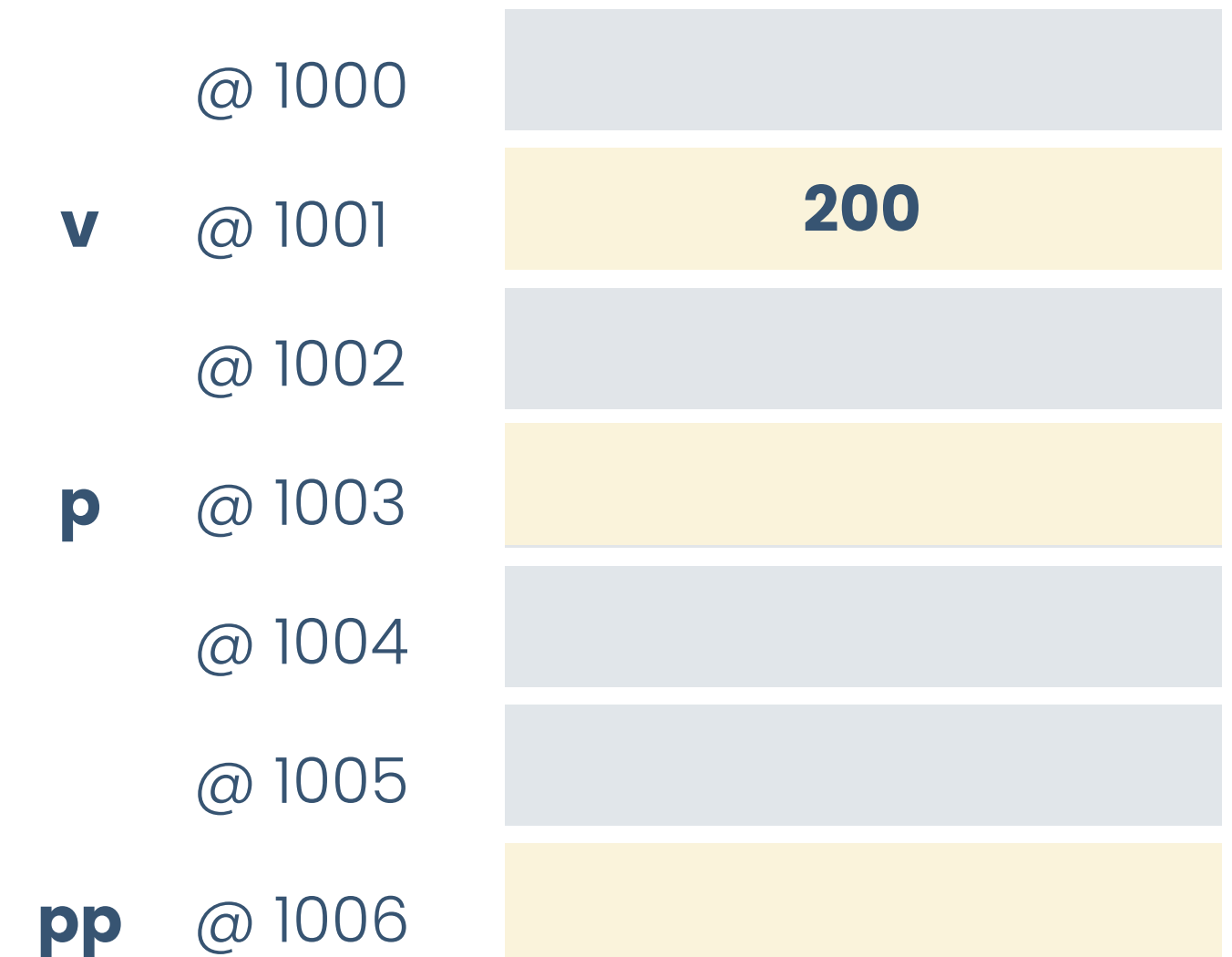
- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  pp : punter_a_punter_a_enter;
fvar
inici
  v := 200;

falgorisme
```

● → Si us hi fixeu, he de seguir especificant quin és el tipus final a on apunto (aquí, enter).



Dobles punters

Un punter també pot apuntar a un altre punter !

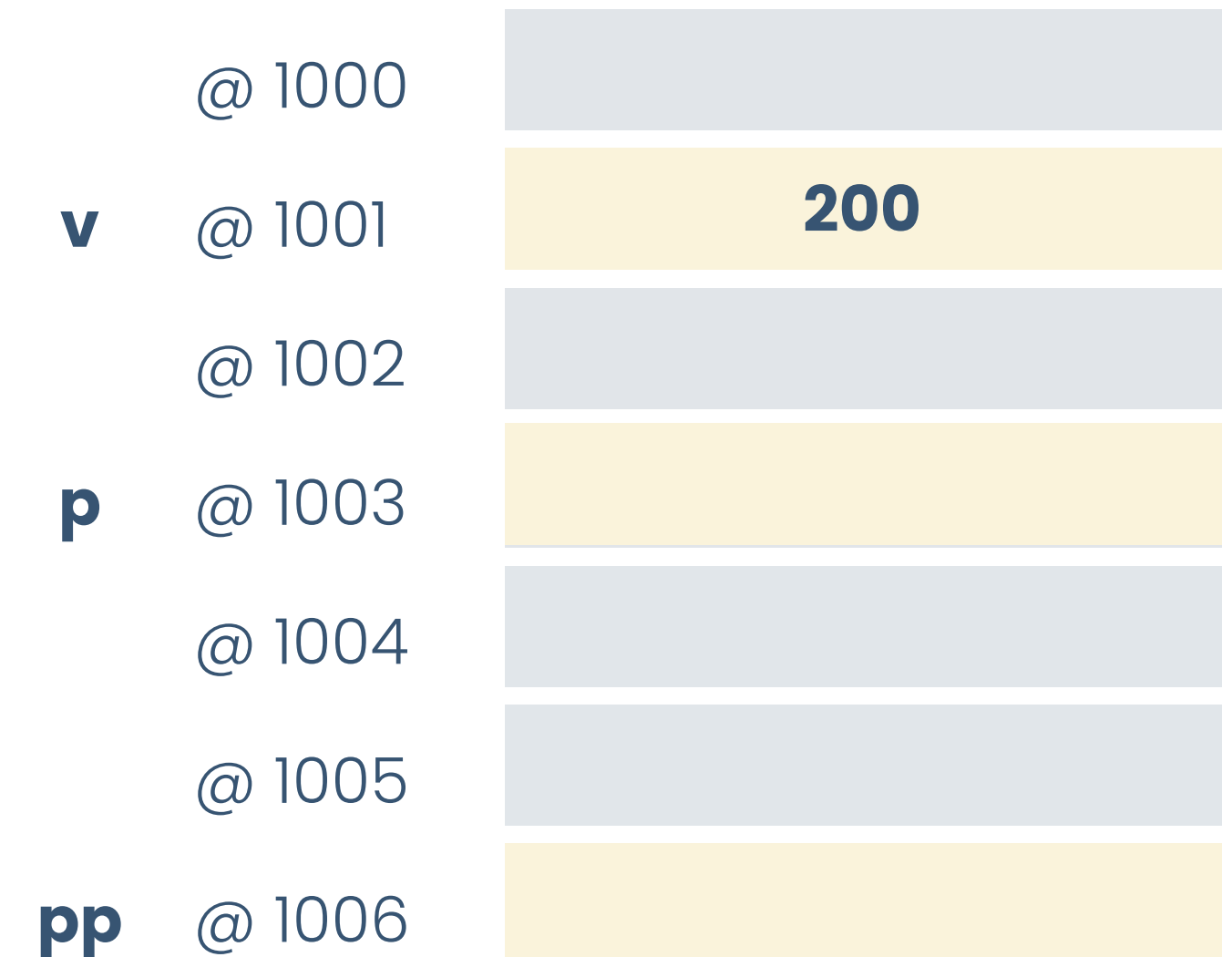
- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  pp : punter_a_punter_a_enter;
fvar
inici
  v := 200;
  p := &v;

falgorisme
```

● → Si us hi fixeu, he de seguir especificant quin és el tipus final a on apunto (aquí, enter).



Dobles punters

Un punter també pot apuntar a un altre punter !

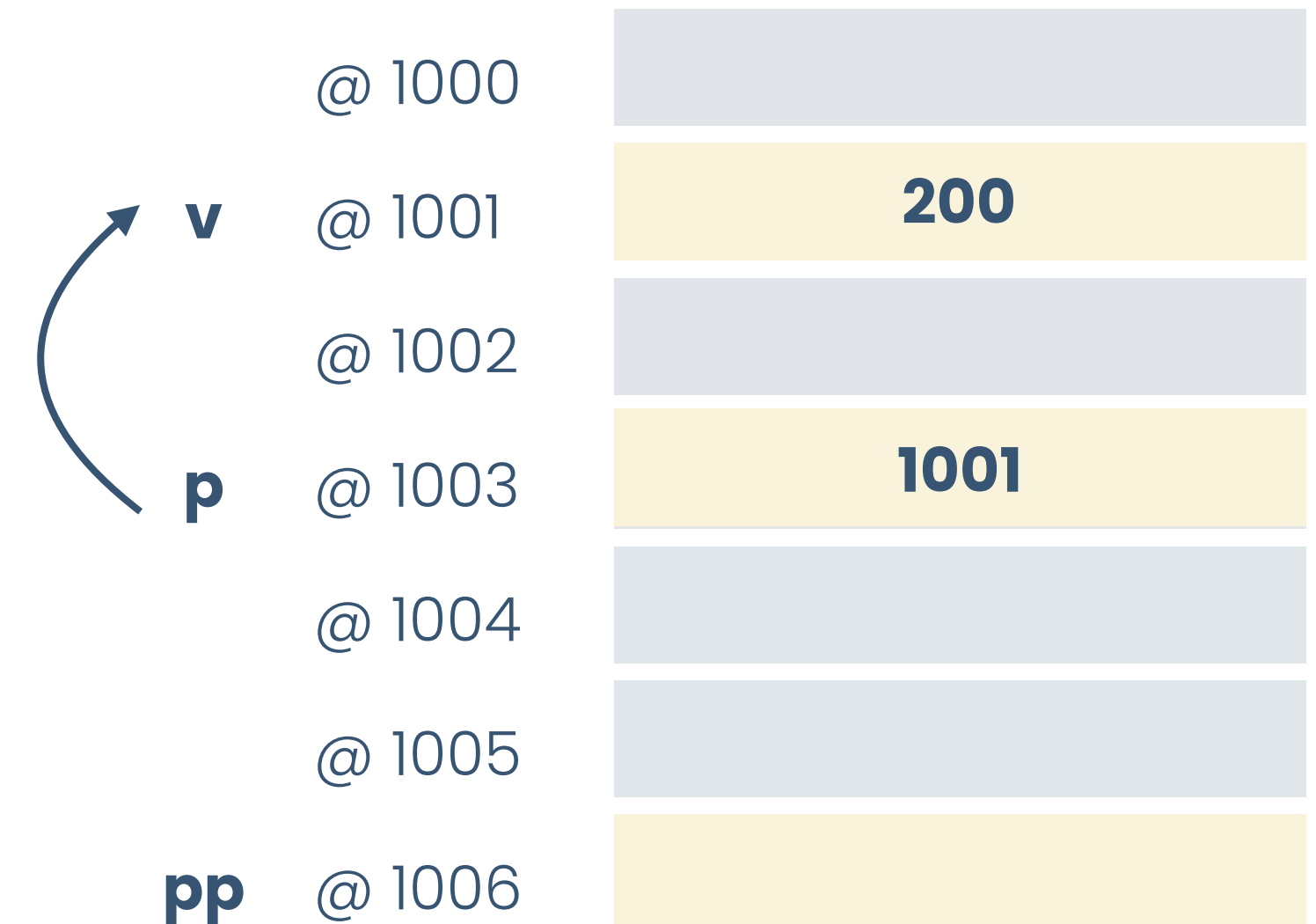
- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  pp : punter_a_punter_a_enter;
fvar
inici
  v := 200;
  p := &v;

falgorisme
```

Si us hi fixeu, he de seguir especificant quin és el tipus final a on apunto (aquí, enter).



Dobles punters

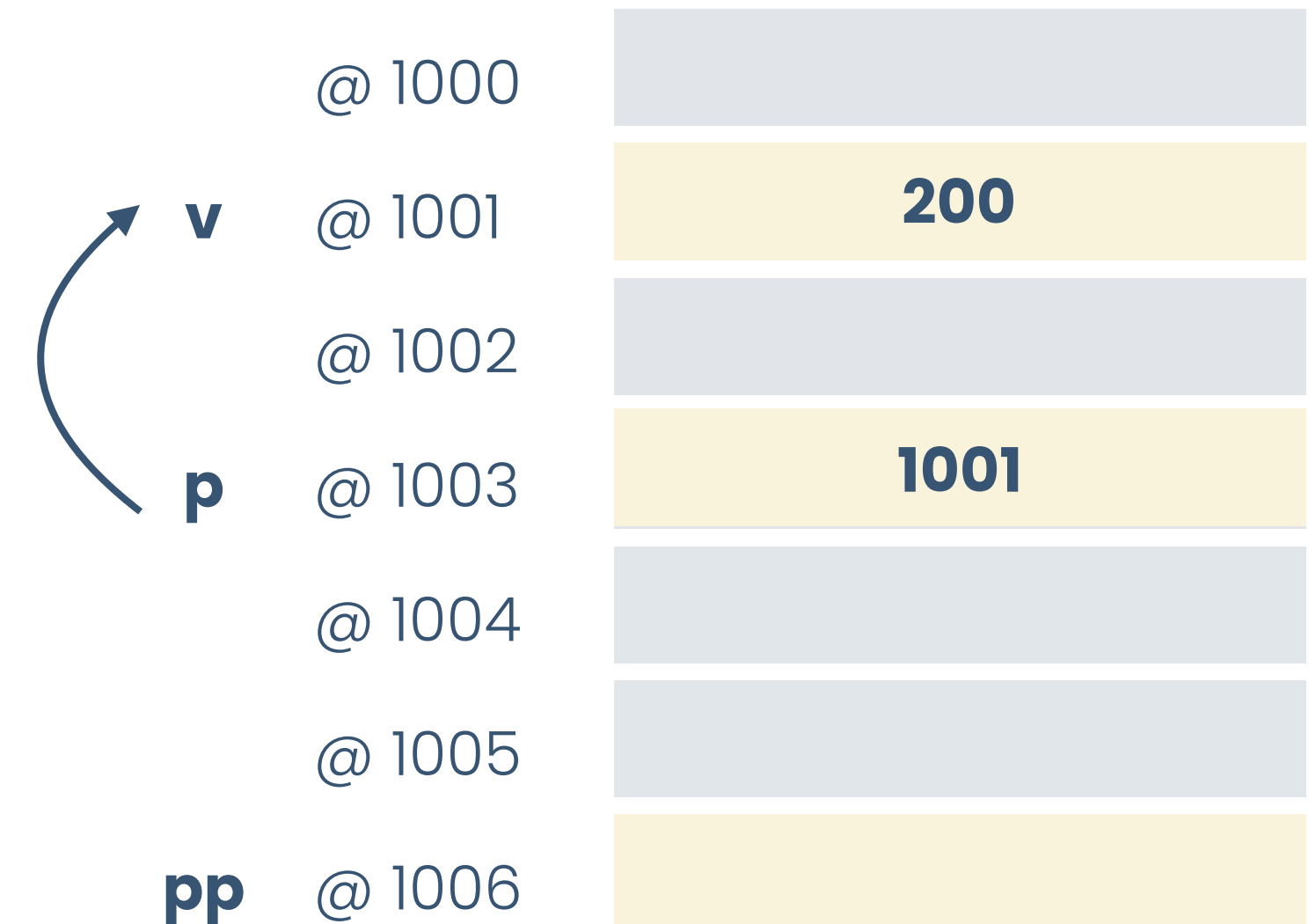
Un punter també pot apuntar a un altre punter !

- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  pp : punter_a_punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
  pp := &p;
falgorisme
```

Si us hi fixeu, he de seguir especificant quin és el tipus final a on apunto (aquí, enter).



Dobles punters

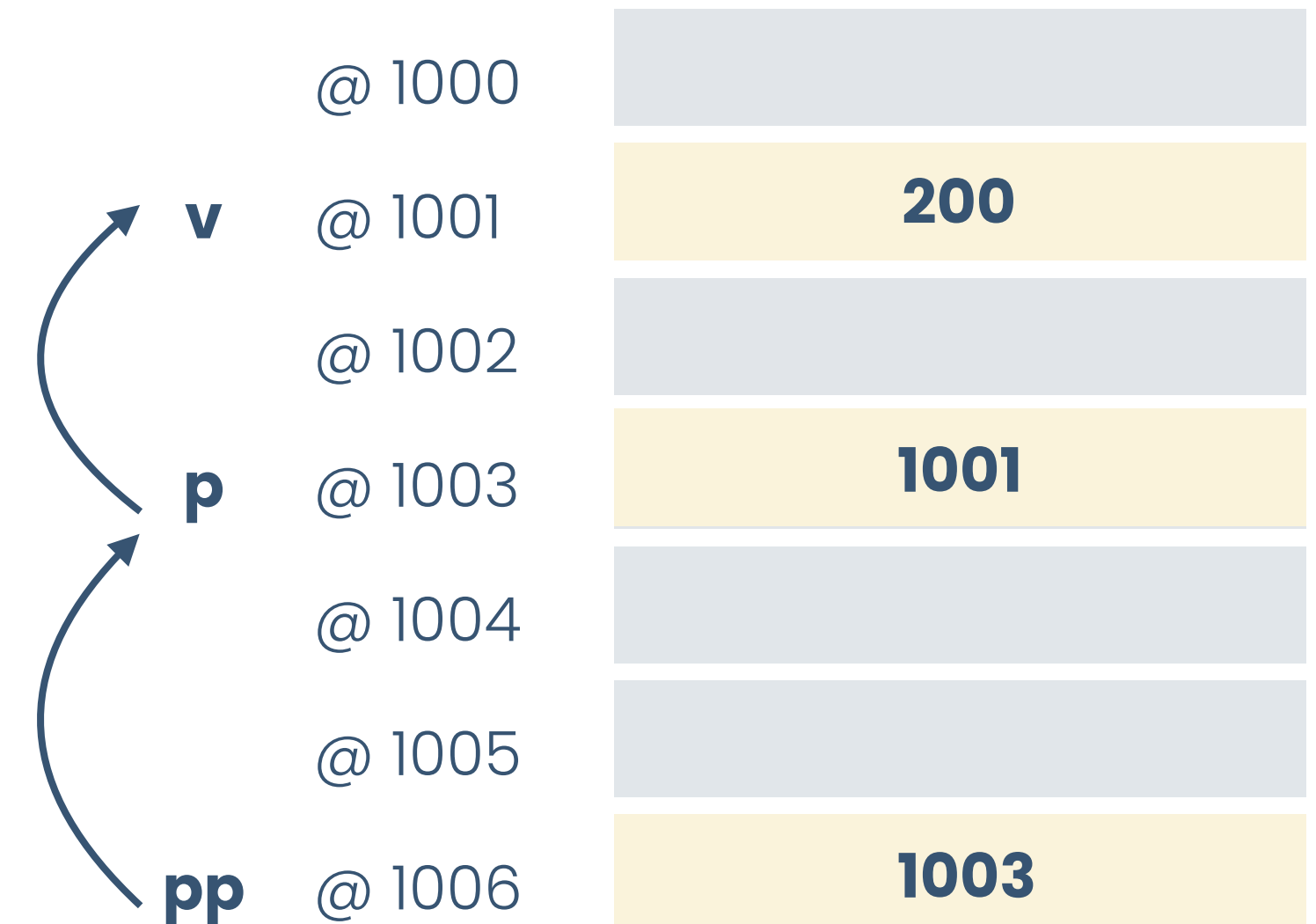
Un punter també pot apuntar a un altre punter !

- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  pp : punter_a_punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
  pp := &p;
falgorisme
```

Si us hi fixeu, he de seguir especificant quin és el tipus final a on apunto (aquí, enter).



Dobles punters

Un punter també pot apuntar a un altre punter !

- Fins ara hem vist com un punter pot apuntar a una variable.
- Però com que un punter no és res més que una variable que guarda una adreça de memòria, res ens impedeix que a aquesta adreça de memòria també hi hagi un punter!
- Un punter doble és un punter que apunta a un altre punter

Funcionament

algorisme exemple és

var

 v : enter;

 p : punter_a_enter;

 pp : punter_a_punter_a_enter;

fvar

inici

 v := 200;

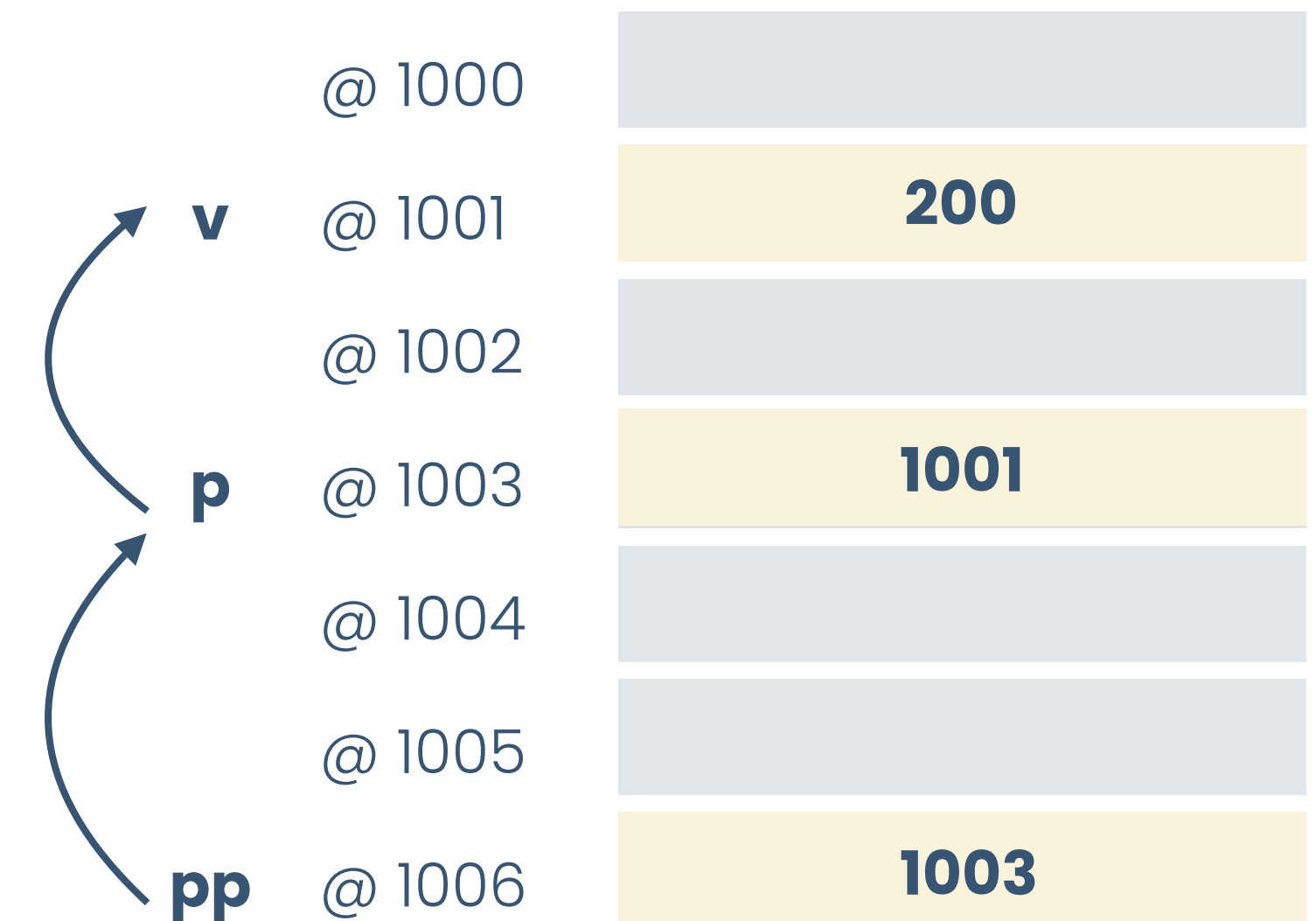
 p := &v;

 pp := &p;

falgorisme

Si us hi fixeu, he de seguir especificant quin és el tipus final a on apunto (aquí, enter).

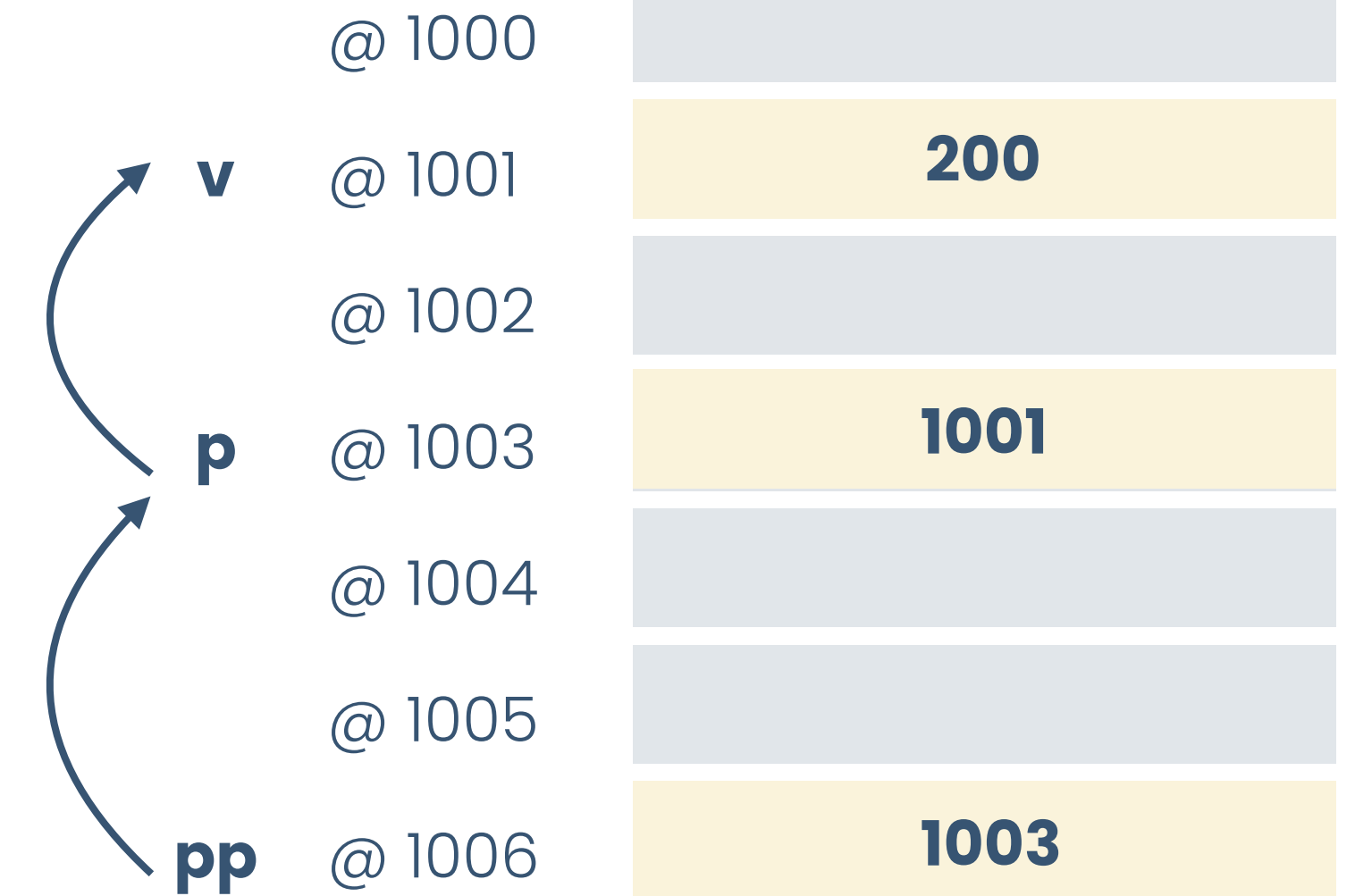
"pp" Apunta a "p", no a "v" !!!!
No és el mateix! Entens per què?



Dobles punters

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  pp : punter_a_punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
  pp := &p;
falgorisme
```



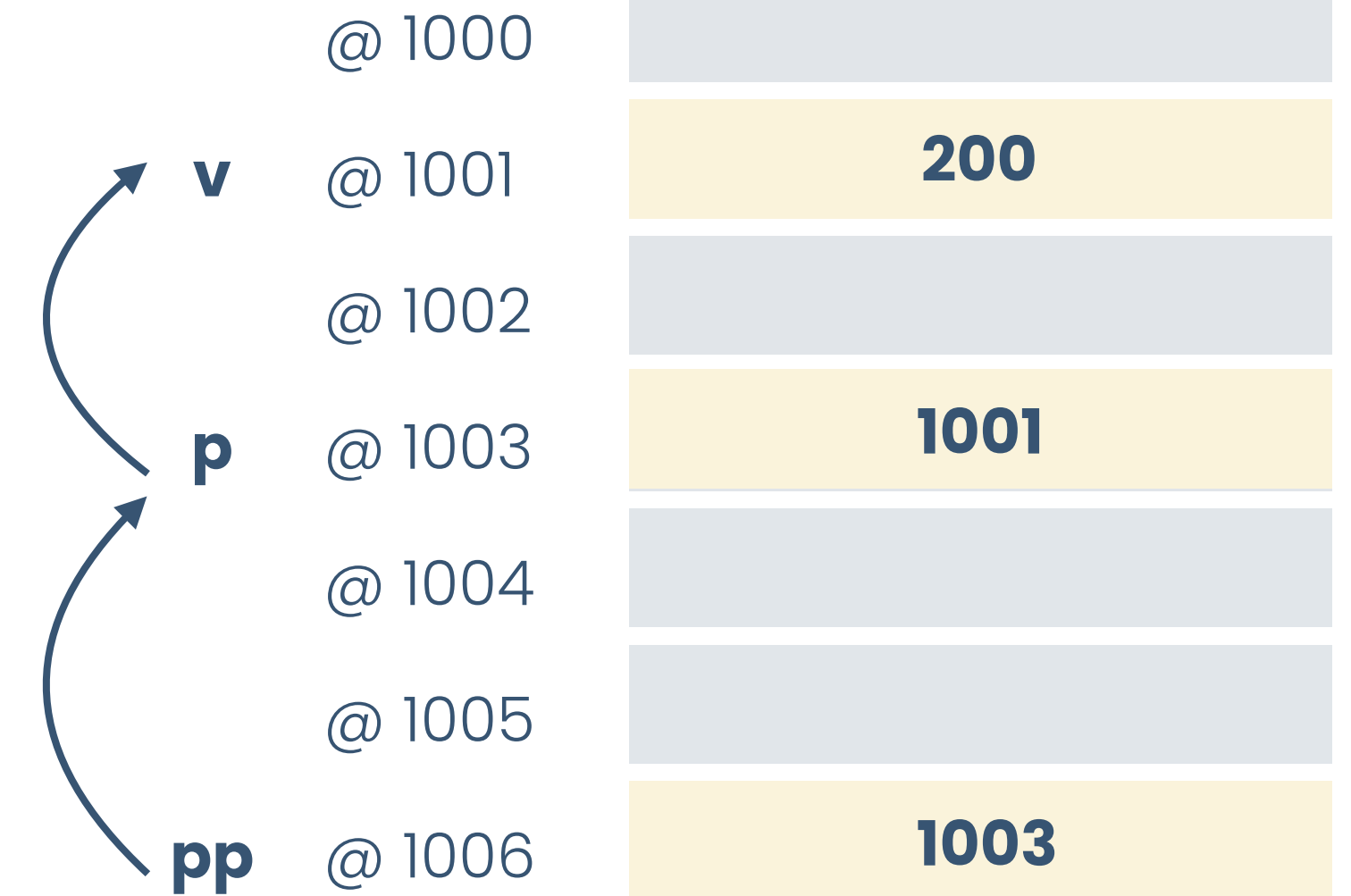
?

- A qui estic fent referència si faig ***pp**?
- A qui estic fent referència si faig ****pp**?
- Com puc modificar **v** a través de **pp**?

Dobles punters

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  pp : punter_a_punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
  pp := &p;
falgorisme
```



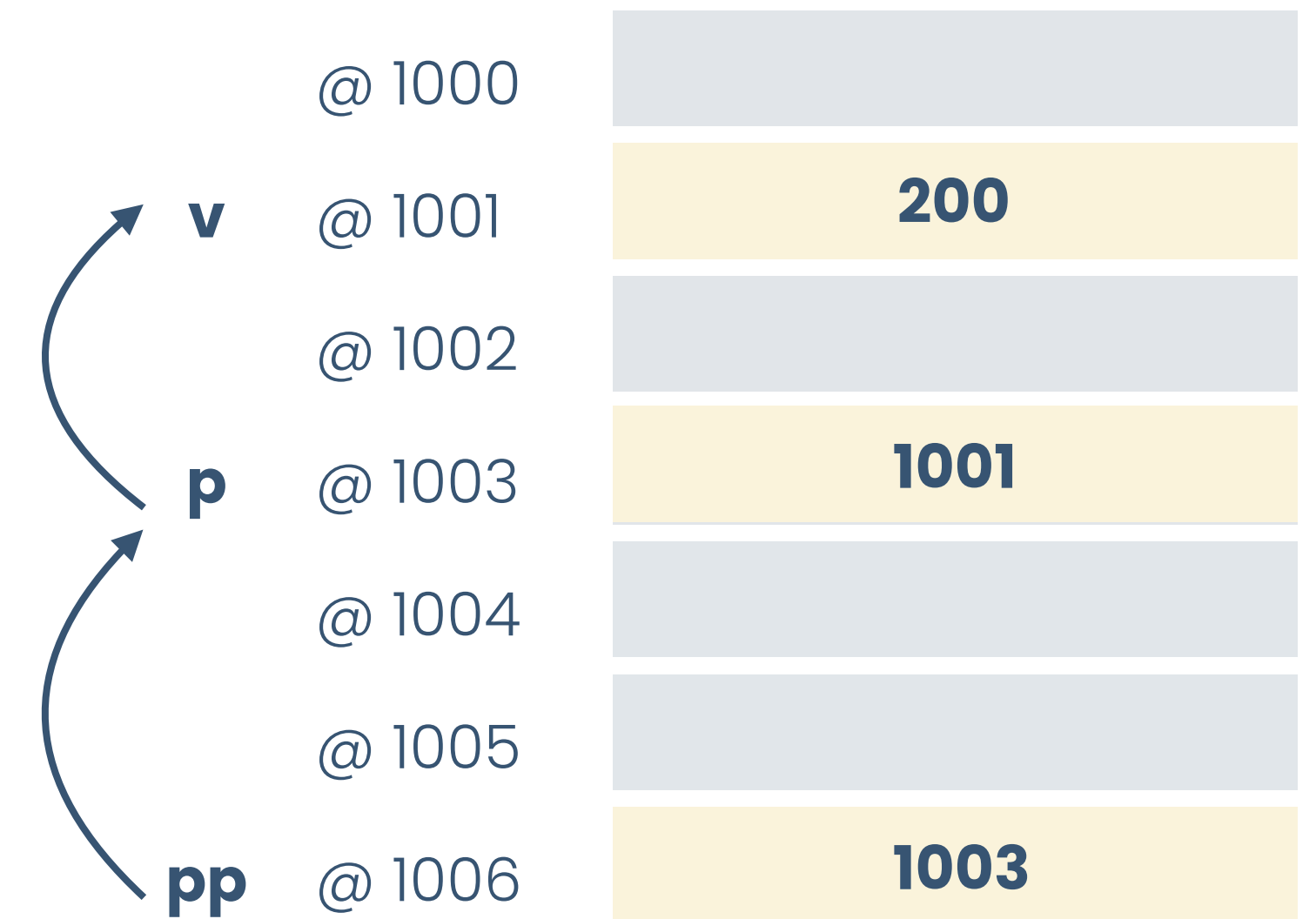
?

- A qui estic fent referència si faig ***pp**?
 - A la variable **p**.
- A qui estic fent referència si faig ****pp**?
- Com puc modificar **v** a través de **pp**?

Dobles punters

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  pp : punter_a_punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
  pp := &p;
falgorisme
```



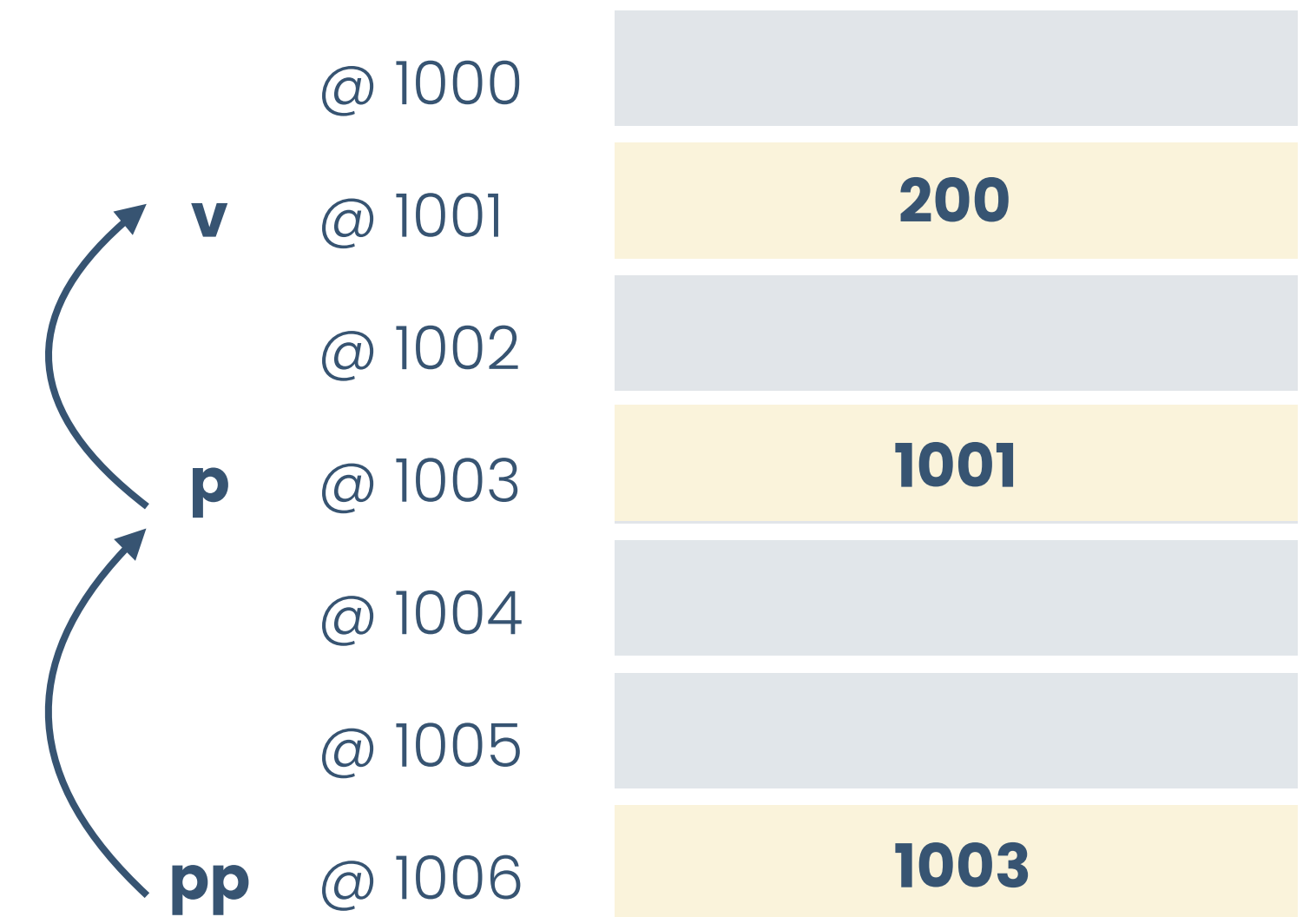
?

- A qui estic fent referència si faig ***pp**?
 - A la variable **p**.
- A qui estic fent referència si faig ****pp**?
 - A la variable **v**.
- Com puc modificar **v** a través de **pp**?

Dobles punters

Funcionament

```
algorisme exemple és
var
  v : enter;
  p : punter_a_enter;
  pp : punter_a_punter_a_enter;
fvar
inici
  v := 200;
  p := &v;
  pp := &p;
falgorisme
```



?

- A qui estic fent referència si faig ***pp**?
 - A la variable **p**.
- A qui estic fent referència si faig ****pp**?
 - A la variable **v**.
- Com puc modificar **v** a través de **pp**?
 - ****pp := 400;**

Dobles punters

Exercici I

- Fes el seguiment en memòria i digues què escriu per pantalla

```
algorisme test és
var
    a, b: enter;
    p: punter_a_enter;
    pp: punter_a_punter_a_enter;
fvar
inici
    a := 3;
    p := &a;
    pp := &p;

    **pp := 10;
    b := a + *p + **pp;
    escriure(" b = ", b);
falgorisme
```

Dobles punters

Exercici II

- Fes el seguiment en memòria i digues què escriu per pantalla

```
algorisme test és
var
  a, b: enter;
  p, q: punter_a_enter;
  pp: punter_a_punter_a_enter;
fvar
inici
  a := 3;
  b := 7;
  p := &a;
  q := &b;
  pp := &p;
  b := b + **pp;
  *p := *q - a;
  escriure("a = ", a, " b = ", b);
falgorisme
```

Dobles punters

Per a què es fan servir els dobles punters?

1

Quan necessitem modificar un punter des de dins d'una funció.

Un punter conté una adreça de memòria. Quan passes un punter a una funció, passes **una còpia d'aquesta adreça**, no el punter original. Això vol dir que:

- Si modifies el contingut apuntat pel punter (*p), el canvi es reflecteix a l'exterior. ✓
- Si intentes modificar l'adreça del punter (p = ...), només estàs modificant la còpia dins la funció, i l'original no canvia. ✗

Dobles punters

Per a què es fan servir els dobles punters?

1

Quan necessitem modificar un punter des de dins d'una funció.

Un punter conté una adreça de memòria. Quan passes un punter a una funció, passes **una còpia d'aquesta adreça**, no el punter original. Això vol dir que:

- Si modifikes el contingut apuntat pel punter (*p), el canvi es reflecteix a l'exterior. ✓
- Si intentes modificar l'adreça del punter (p = ...), només estàs modificant la còpia dins la funció, i l'original no canvia. ✗

```
#include <stdio.h>
```

```
void canvia_punter(int *p) {  
    int x = 50;  
    p = &x; // Aquí canviem el punter, però només la còpia local  
}
```

```
int main() {  
    int a = 10;  
    int *p = &a;  
  
    canvia_punter(p); // Intentem modificar p  
    printf("Valor de p després de la funció: %d\n", *p);  
    // Continua sent 10  
  
    return 0;  
}
```



Dobles punters


Per a què es fan servir els dobles punters?

1

Quan necessitem modificar un punter des de dins d'una funció.

Un punter conté una adreça de memòria. Quan passes un punter a una funció, passes **una còpia d'aquesta adreça**, no el punter original. Això vol dir que:

- Si modifies el contingut apuntat pel punter (*p), el canvi es reflecteix a l'exterior. ✓
- Si intentes modificar l'adreça del punter (p = ...), només estàs modificant la còpia dins la funció, i l'original no canvia. ✗
- Amb un punter doble sí que podrem modificar-lo.



```
#include <stdio.h>

void canvia_punter(int **p) {
    int x = 50;
    *p = &x; // Modifiquem l'original a través del punter doble
}

int main() {
    int a = 10;
    int *p = &a;

    canvia_punter(&p); // Passem l'adreça de 'p'
    printf("Valor de p després de la funció: %d\n", *p);
    // Ara apunta a 'x'

    return 0;
}
```


Dobles punters

Per a què es fan servir els dobles punters?

1

Quan necessitem modificar un punter des de dins d'una funció.

Un punter conté una adreça de memòria. Quan passes un punter a una funció, passes **una còpia d'aquesta adreça**, no el punter original. Això vol dir que:

- Si modifies el contingut apuntat pel punter (*p), el canvi es reflecteix a l'exterior. ✓
- Si intentes modificar l'adreça del punter (p = ...), només estàs modificant la còpia dins la funció, i l'original no canvia. ✗
- Amb un punter doble sí que podrem modificar-lo.

2

Quan fem servir matrius alocatades dinàmicament

- Per definir matrius de 2 dimensions fent servir memòria dinàmica, farem servir dobles punters.
- Si necessitem modificar una matriu dinàmica des de dins d'una funció, també necessitarem dobles punters.

➡ Ho veurem al tema següent

Dobles punters

Per a què es fan servir els dobles punters?

1

Quan necessitem modificar un punter des de dins d'una funció.

Un punter conté una adreça de memòria. Quan passes un punter a una funció, passes **una còpia d'aquesta adreça**, no el punter original. Això vol dir que:

- Si modifies el contingut apuntat pel punter (*p), el canvi es reflecteix a l'exterior. ✓
- Si intentes modificar l'adreça del punter (p = ...), només estàs modificant la còpia dins la funció, i l'original no canvia. ✗
- Amb un punter doble sí que podrem modificar-lo.

2

Quan fem servir matrius alocatades dinàmicament

- Per definir matrius de 2 dimensions fent servir memòria dinàmica, farem servir dobles punters.
- Si necessitem modificar una matriu dinàmica des de dins d'una funció, també necessitarem dobles punters.

➡ Ho veurem al tema següent

3

Per gestionar estructures de dades

- A vegades, en lloc de guardar un únic valor, volem guardar-ne molts i gestionar-los de manera flexible. Per exemple, podríem tenir una sèrie de valors enllaçats entre ells.
- Per fer això, fem servir punters que apunten a altres punters.

➡ Ho veureu a l'assignatura d'Estructures de Dades

EXERCICIS

Exercicis

1. Modificar un valor a través d'un punter (facilíssim):

Escriu una funció, anomenada `incrementa`, que rebi un enter per referència i li sumi 1 unitat.

2. Modificar un valor a través d'un punter (II) (facilíssim):

Escriu una funció `canvia_signe` que inverteixi el signe d'un nombre passat per referència.

3. Intercanvi de tres valors (facilíssim)

Escriu una funció, anomenada `intercanvi`, que rebi tres enters (`a`, `b` i `c`) i faci que `a` valgui `b`, `b` valgui `a`, i `c` valgui la suma de `a` + `b`.

4. Trobar el màxim de 2 nombres (facilíssim)

Escriu una funció `maxim` que rebi dos nombres i guardi el màxim en un tercer paràmetre per referència.

5. Comptar xifres (fàcil)

Escriu una funció `compta_xifres` que donat un nombre, en compti quantes xifres té i ho retorni mitjançant un paràmetre per referència.

6. Quocient i residu (fàcil)

Escriu una funció, anomenada `divisio` que rebi dos enters i guardi el quocient i el residu en dos paràmetres passats per referència.

7. Comptar quantes vegades apareix un nombre en un array (fàcil)

Escriu una funció `compta_aparicions` que rebi un vector d'enters i la seva mida, un valor a buscar i un punter on guardarà quantes vegades apareix aquest valor.

8. Retornar el nombre més gran i el més petit d'un array (mig)

Escriu una funció `min_max` que rebi un array d'enters i la seva mida, i retorni el valor mínim i màxim per referència.

9. Trobar el segon nombre més gran d'un array (mig+)

Escriu una funció `segon_mes_gran` que trobi el segon nombre més gran en un array d'enters i el retorni per referència. La funció no pot modificar l'array original.



Un petit avanç de la classe de laboratoris

Que tracta del motiu pel qual la gent es fa un embolic amb els punters

Punters en C

Declaració d'un punter

PSEUDOCODI:

```
var  
    i: punter_a_enter;  
fvar
```

En C:

```
int *i;
```

Punters en C

PSEUDOCODI:

Declaració d'un punter

```
var  
    i: punter_a_enter;  
fvar
```

**Fer que un punter
apunti a una variable
"a"**

```
i := &a;
```

En C:

```
int *i;
```

```
i = &a;
```

Punters en C

	PSEUDOCODI:	En C:
Declaració d'un punter	<pre>var i: punter_a_enter; fvar</pre>	<pre>int *i;</pre>
Fer que un punter apunti a una variable "a"	<pre>i := &a;</pre>	<pre>i = &a;</pre>
Accedir als continguts de "a" a través del punter	<pre>*i := 50;</pre>	<pre>*i = 50;</pre>

La confusió en la sintaxi dels punters

La confusió en la sintaxi dels punters

- Un dels problemes més grans que hi ha amb els punters en C, no ve de que la gent no entengui el concepte de punter, sinó de la terminologia que es fa servir.
- El símbol 'asterisc' es fa servir per a dues coses diferents, cosa que viola tots els principis del bon disseny (*coses que tenen funcions dràsticament diferents no haurien d'assemblar-se!*). És una mala decisió que es va prendre quan es va dissenyar C, amb la qual ara hi hem de conviure.
- Per aquest motiu, has d'intentar recordar el següent:

La confusió en la sintaxi dels punters

- Un dels problemes més grans que hi ha amb els punters en C, no ve de que la gent no entengui el concepte de punter, sinó de la terminologia que es fa servir.
- El símbol 'asterisc' es fa servir per a dues coses diferents, cosa que viola tots els principis del bon disseny (*coses que tenen funcions dràsticament diferents no haurien d'assemblar-se!*). És una mala decisió que es va prendre quan es va dissenyar C, amb la qual ara hi hem de conviure.
- Per aquest motiu, has d'intentar recordar el següent:

**Un asterisc en una declaració significa que
la variable és un punter**

**Un asterisc fora d'una declaració significa
accedir al valor**

La confusió en la sintaxi dels punters

Un asterisc en una declaració significa que la variable és un punter

Un asterisc fora d'una declaració significa accedir al valor

```
#include <stdio.h>
```

```
int main( )
```

```
{
```

```
    int a = 100;
```

```
    int *x; → Aquest asterisc fa referència a un punter
```

```
    x = &a;
```

```
    *x = 200; → Aquest asterisc fa referència a un enter
```

```
    return 0;
```

```
}
```

La confusió en la sintaxi dels punters

- També pots intentar recordar-ho d'una altra manera:

La confusió en la sintaxi dels punters

- També pots intentar recordar-ho d'una altra manera:
- Oi que quan fem: `char c;` o `float num;` ...

La confusió en la sintaxi dels punters

- També pots intentar recordar-ho d'una altra manera:
- Oi que quan fem: `char c;` o `float num;` ...

... el que estem dient és que: `char c;`

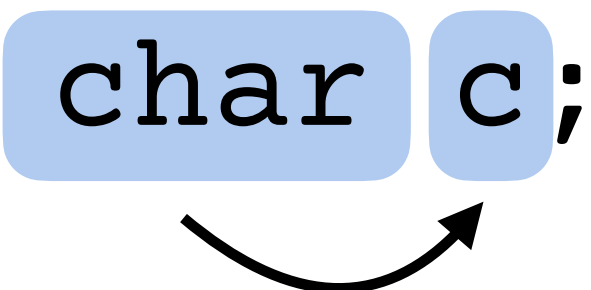


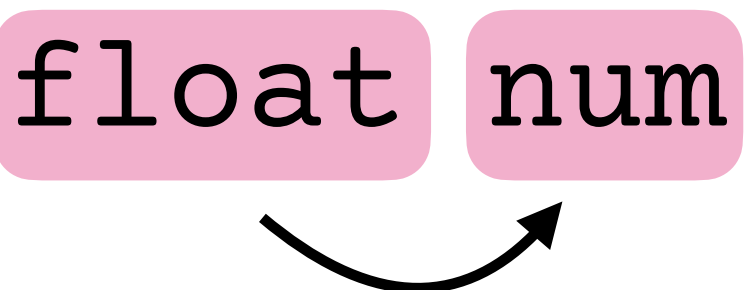
Com que char vol dir caràcter,
c és un caràcter.

La confusió en la sintaxi dels punters

- També pots intentar recordar-ho d'una altra manera:
- Oi que quan fem: `char c;` o `float num;` ...

... el que estem dient és que:

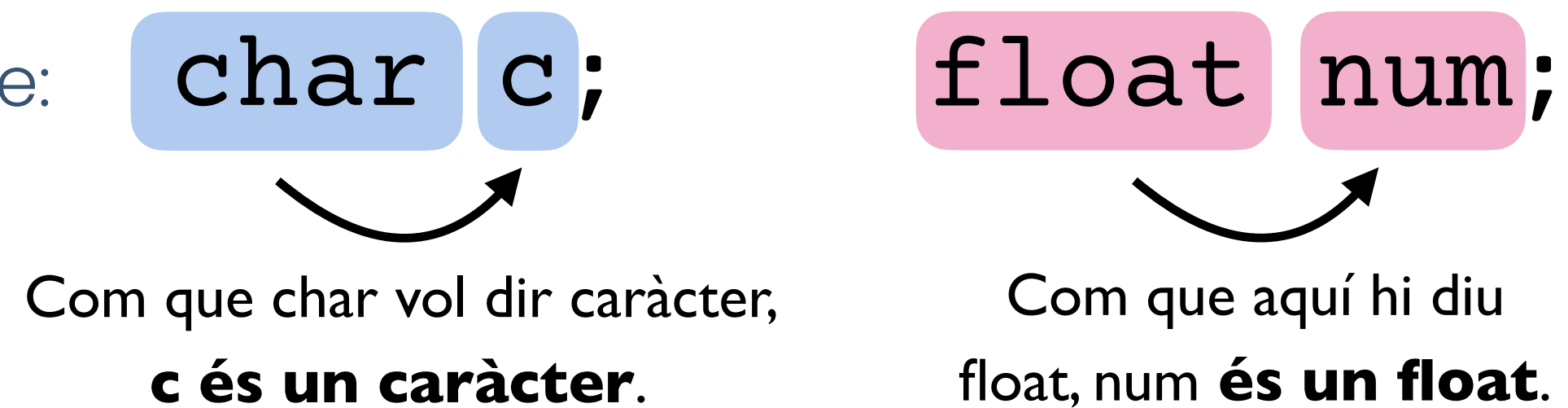
`char c;`

Com que char vol dir caràcter,
c és un caràcter.

`float num;`

Com que aquí hi diu
float, num **és un float.**

La confusió en la sintaxi dels punters

- També pots intentar recordar-ho d'una altra manera:
- Oi que quan fem: `char c;` o `float num;` ...

... el que estem dient és que:

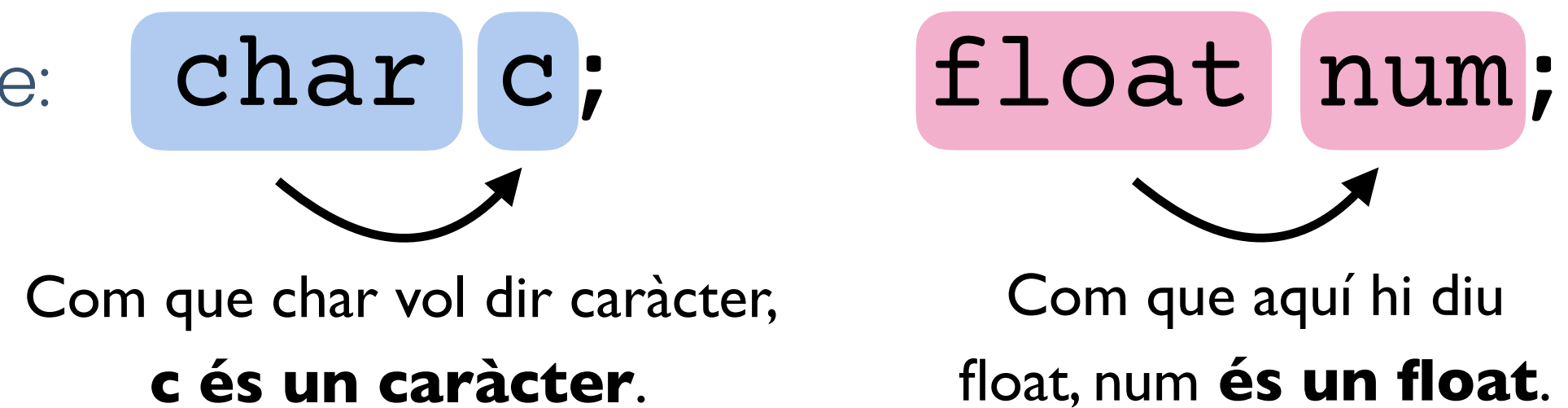


- Doncs el truc per recordar quan fem referència a un punter o als continguts als quals apunta és mirar-se una declaració de punter de la mateixa manera:

La confusió en la sintaxi dels punters

- També pots intentar recordar-ho d'una altra manera:
- Oi que quan fem: `char c;` o `float num;` ...

... el que estem dient és que:



- Doncs el truc per recordar quan fem referència a un punter o als continguts als quals apunta és mirar-se una declaració de punter de la mateixa manera:

```
int*   x   ;
```

La confusió en la sintaxi dels punters

- També pots intentar recordar-ho d'una altra manera:
- Oi que quan fem: `char c;` o `float num;` ...

... el que estem dient és que:

`char c;`
Com que char vol dir caràcter,
c és un caràcter.

`float num;`
Com que aquí hi diu
float, num **és un float.**

- Doncs el truc per recordar quan fem referència a un punter o als continguts als quals apunta és mirar-se una declaració de punter de la mateixa manera:

`int* x;`

Això vol dir un
punter a enter,
oi?

La confusió en la sintaxi dels punters

- També pots intentar recordar-ho d'una altra manera:
- Oi que quan fem: `char c;` o `float num;` ...

... el que estem dient és que:

`char c;`
Com que char vol dir caràcter,
c és un caràcter.

`float num;`
Com que aquí hi diu
float, num **és un float.**

- Doncs el truc per recordar quan fem referència a un punter o als continguts als quals apunta és mirar-se una declaració de punter de la mateixa manera:

`int* x;`
Això vol dir un punter a enter, oi?
Doncs això també!
Quan que em trobi la "x" sola, estaré fent referència a un punter a enter!

La confusió en la sintaxi dels punters

- També pots intentar recordar-ho d'una altra manera:
- Oi que quan fem: `char c;` o `float num;` ...

... el que estem dient és que:

`char c;`

Com que char vol dir caràcter,
c és un caràcter.

`float num;`

Com que aquí hi diu
float, num **és un float.**

- Doncs el truc per recordar quan fem referència a un punter o als continguts als quals apunta és mirar-se una declaració de punter de la mateixa manera:

- *En canvi, si m'ho miro d'aquesta altra manera:*

`int* x;`

Això vol dir un
punter a enter,
oi?

Doncs això també!

Quan que em trobi la "x"
sola, estaré fent referència a
un punter a enter!

`int *x ;`

La confusió en la sintaxi dels punters

- També pots intentar recordar-ho d'una altra manera:
- Oi que quan fem: `char c;` o `float num;` ...

... el que estem dient és que:

`char c;`

Com que char vol dir caràcter,
c és un caràcter.

`float num;`

Com que aquí hi diu
float, num **és un float.**

- Doncs el truc per recordar quan fem referència a un punter o als continguts als quals apunta és mirar-se una declaració de punter de la mateixa manera:

- *En canvi, si m'ho miro d'aquesta altra manera:*

`int* x;`

Això vol dir un
punter a enter,
oi?

Doncs això també!

Quan que em trobi la "x"
sola, estaré fent referència a
un punter a enter!

`int *x;`

Això és un
enter, oi?

La confusió en la sintaxi dels punters

- També pots intentar recordar-ho d'una altra manera:
- Oi que quan fem: `char c;` o `float num;` ...

... el que estem dient és que:

`char c;`

Com que char vol dir caràcter,
c és un caràcter.

`float num;`

Com que aquí hi diu
float, num **és un float.**

- Doncs el truc per recordar quan fem referència a un punter o als continguts als quals apunta és mirar-se una declaració de punter de la mateixa manera:

- *En canvi, si m'ho miro d'aquesta altra manera:*

`int* x;`

Això vol dir un
punter a enter,
oi?

Doncs això també!
Quan que em trobi la "x"
sola, estaré fent referència a
un punter a enter!

`int *x;`

Això és un
enter, oi?

Doncs això també!
Quan em trobi la "x" amb un
asterisc, estaré fent
referència a un enter (és a
dir, als seus continguts!)



Tasques a fer:

Repassar molt bé aquesta lliçó

Fer els exercicis de teoria

Preparar molt bé el material de laboratoris

Intentar traduir els exercicis de teoria a C

Preguntar si no s'entén!