TEORIA

PROGRAMACIÓ CIENTÍFICA

T7

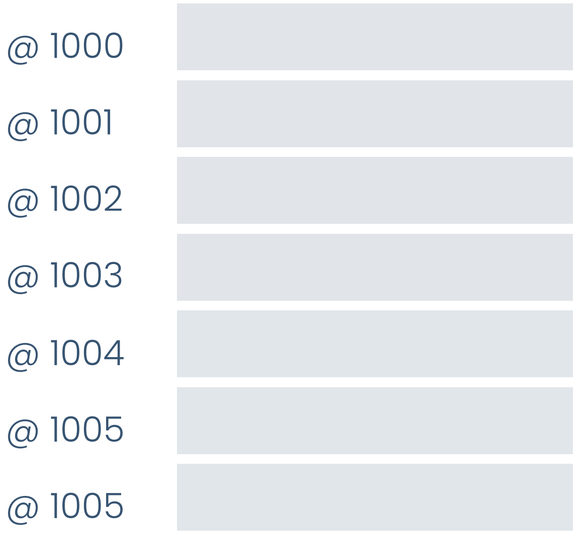
Punters i Pas per referència

PUNTERS

Punters

Funcionament

```
algorisme exemple és
var



fvar
inici


falgorisme
```

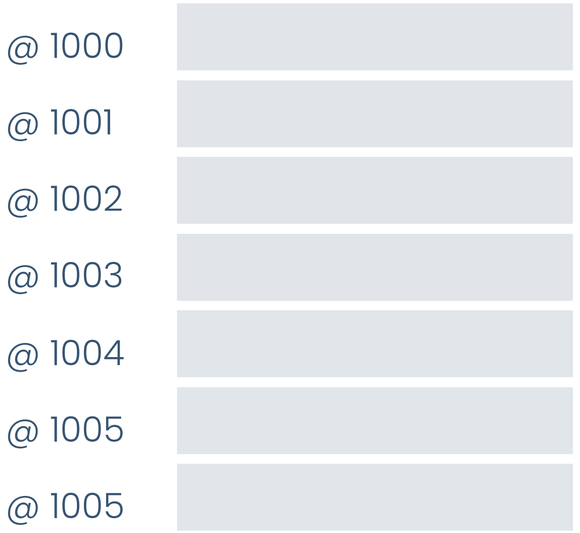


Memòria de 32 bits

Punters

Funcionament

```
algorisme exemple és
var
   v : enter;

fvar
inici


falgorisme
```

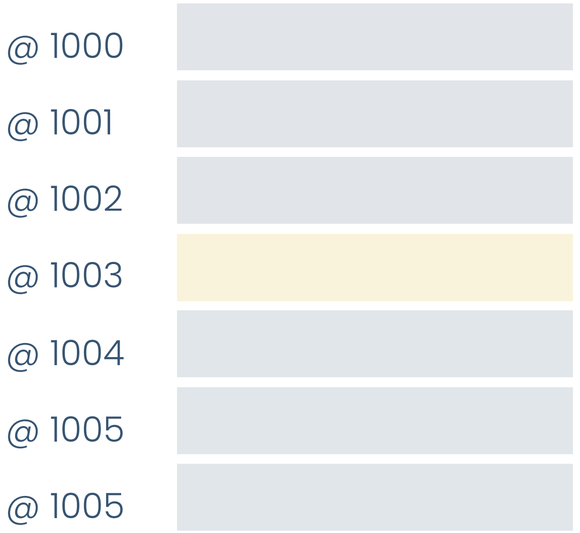


Memòria de 32 bits

Punters

Funcionament

```
algorisme exemple és
var
   v : enter;

fvar
inici


falgorisme
```

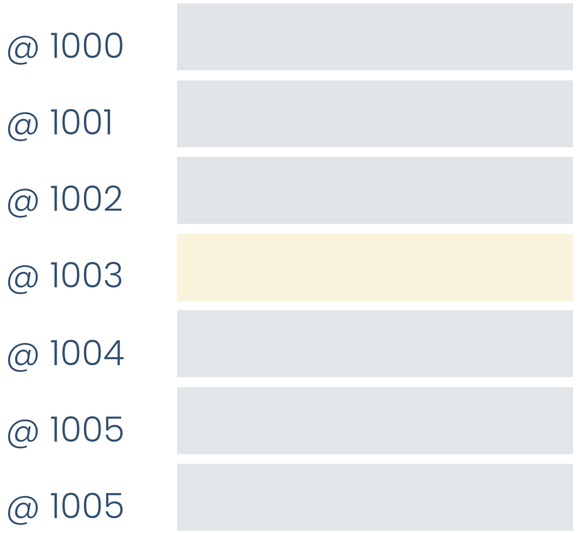


Memòria de 32 bits

Punters

Funcionament

```
algorisme exemple és
var
   v : enter;

fvar
inici
   v := 200;

falgorisme
```

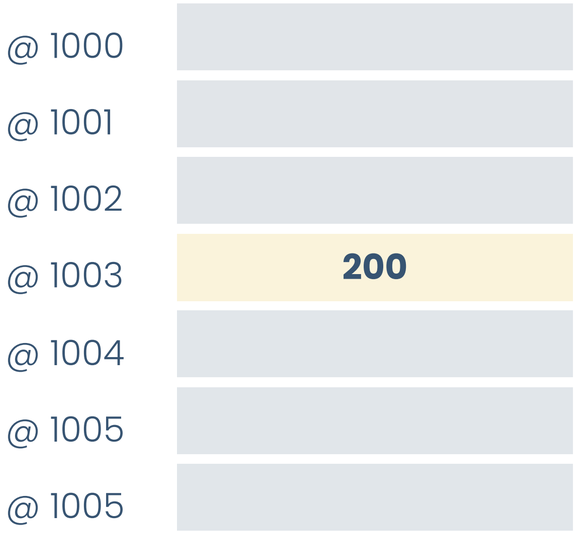


Memòria de 32 bits

Punters

Funcionament

```
algorisme exemple és
var
   v : enter;

fvar
inici
   v := 200;

falgorisme
```

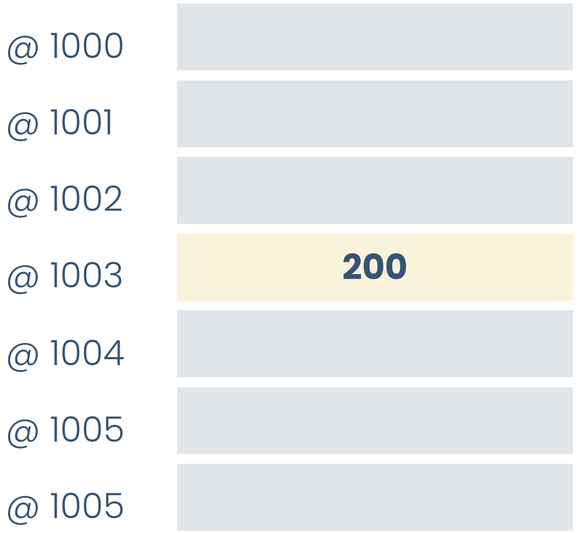


Memòria de 32 bits

Punters

Funcionament

```
algorisme exemple és
var
   v : enter;

fvar
inici
   v := 200;

falgorisme
```



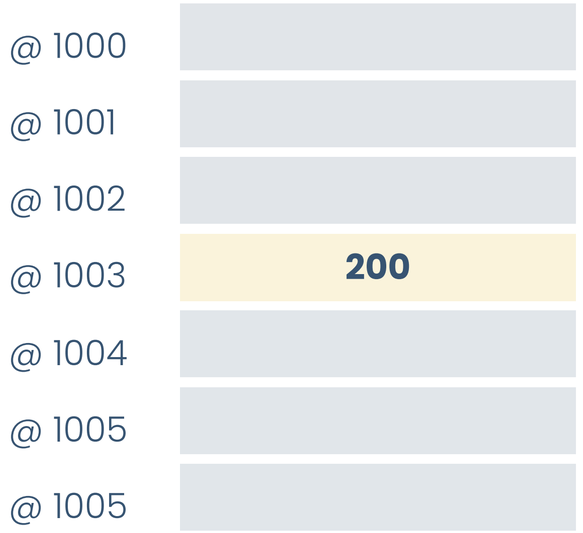
Memòria de 32 bits

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
```



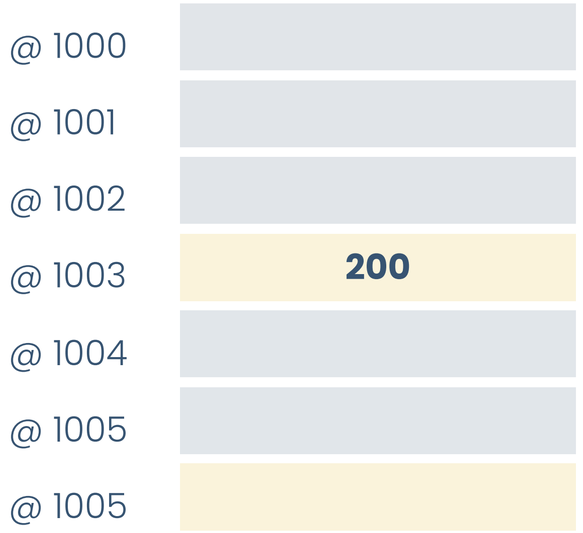
Memòria de 32 bits

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
```



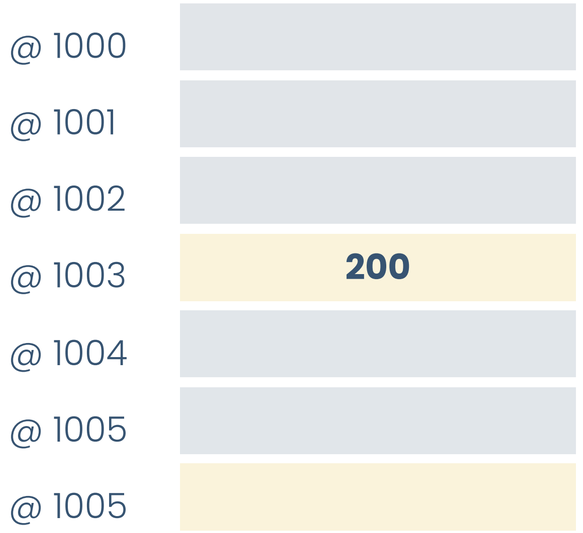
Memòria de 32 bits

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
```



Memòria de 32 bits

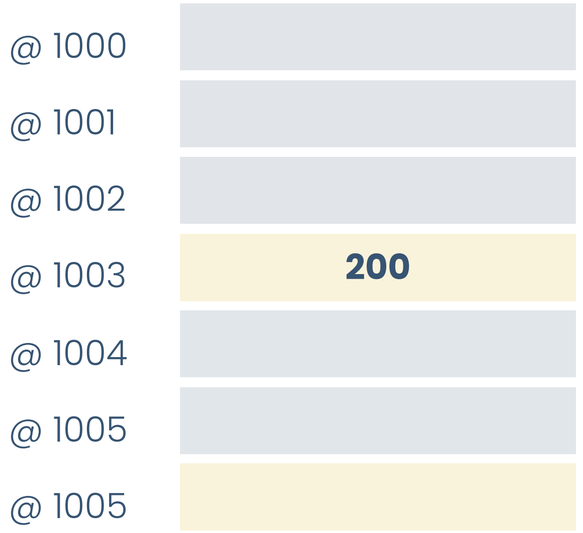
Un punter/apuntador és una variable que conté l'adreça de memòria d'una altra variable.

- 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
```



Memòria de 32 bits

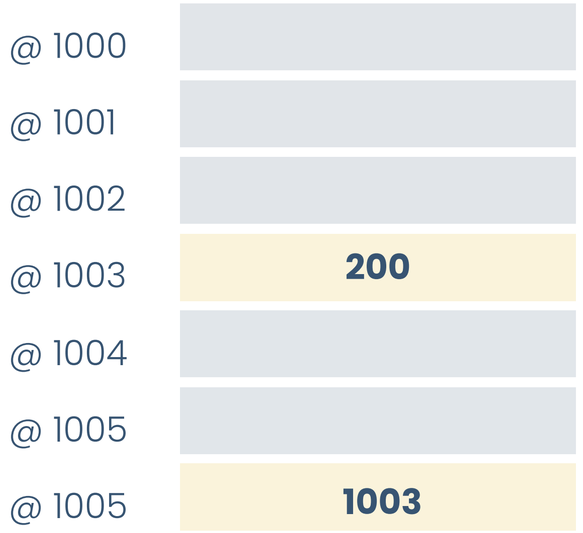
Un punter/apuntador és una variable que conté l'adreça de memòria d'una altra variable.

- 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
```



Memòria de 32 bits

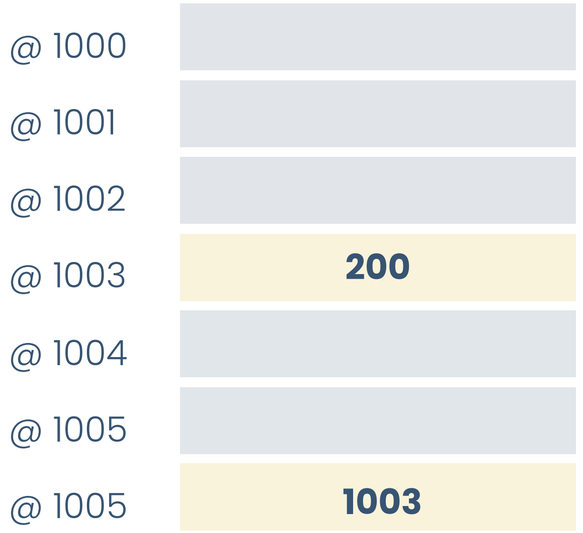
Un punter/apuntador és una variable que conté l'adreça de memòria d'una altra variable.

- 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
```



Memòria de 32 bits

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

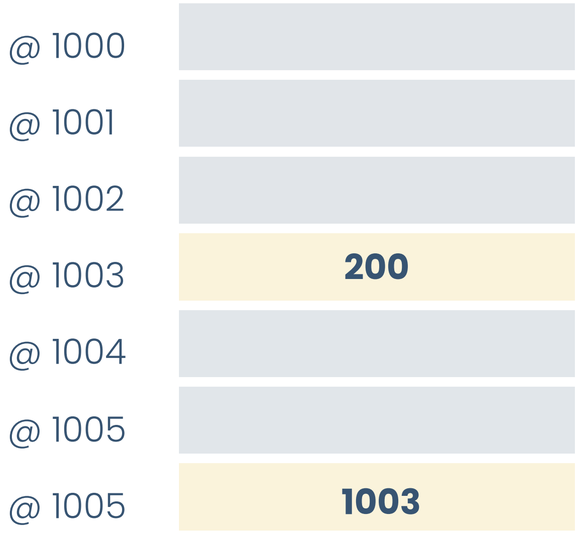
Ús a teoria

```
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?



Memòria de 32 bits

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:

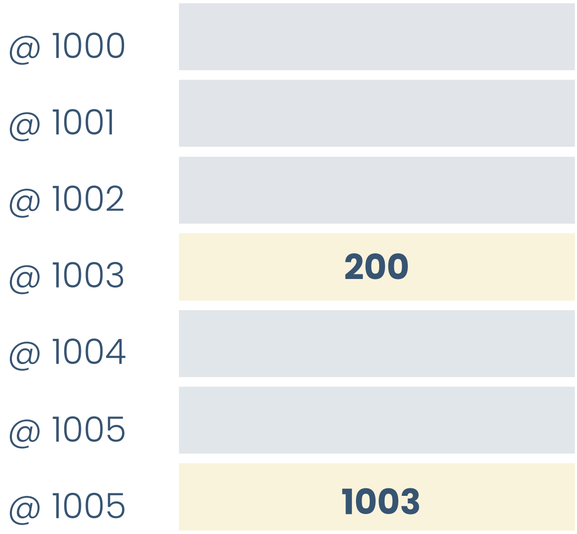
punter_a_enter
punter_a_caracter
punter_a_real

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`** ?



Memòria de 32 bits

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

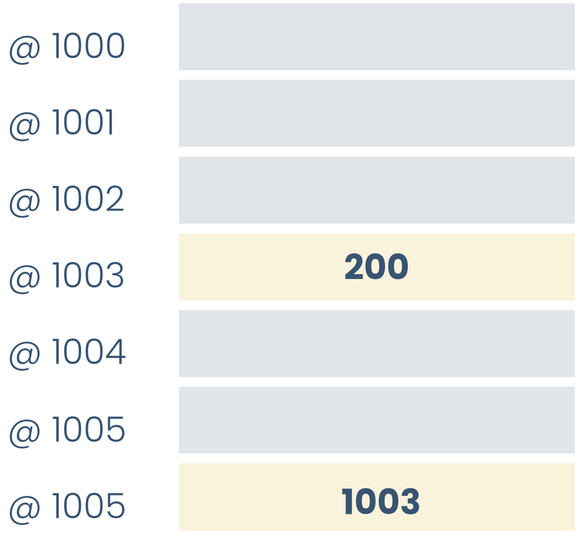
No té sentit, però, declarar punters a taules, recordeu per què?

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 (1005)



Memòria de 32 bits

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?

Memòria de 32 bits

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 (1005)

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`** ?

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?

Memòria de 32 bits

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 (1005)

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

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

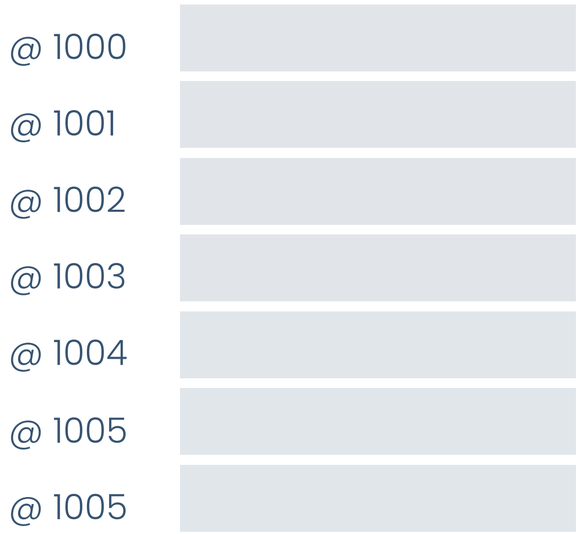
Però què ens donaria fer **`*v`** ?

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

Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var




fvar
inici









falgorisme
```

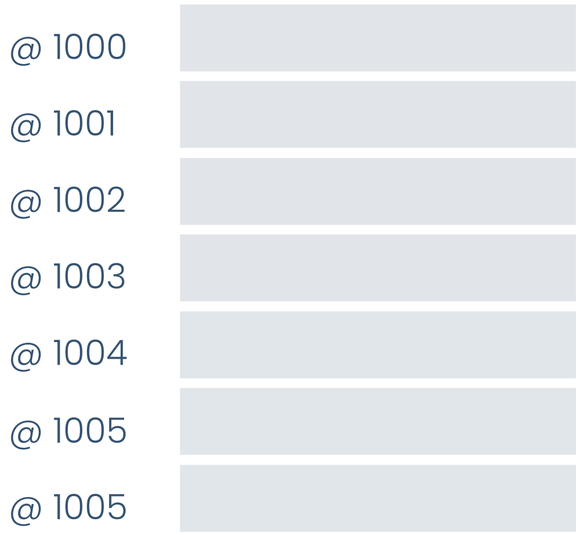


Memòria de 32 bits

Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
   x : enter;



fvar
inici









falgorisme
```

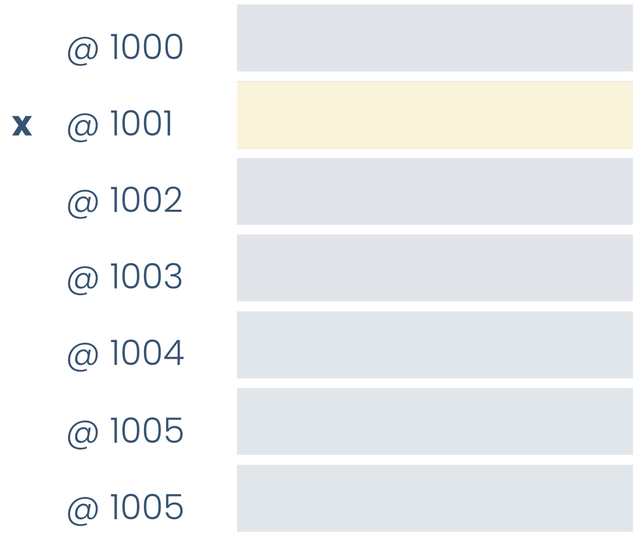


Memòria de 32 bits

Punters

Funcionament: exemple pas a pas

```
algorisme exemple_2 és
var
   x : enter;



fvar
inici









falgorisme
```

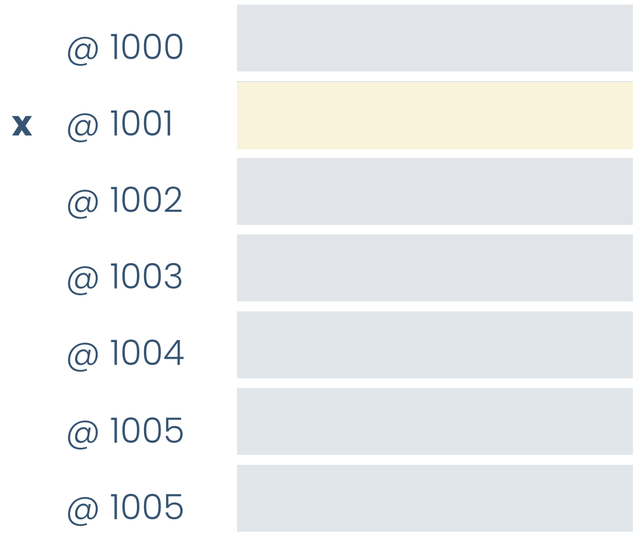


Memòria de 32 bits

Punters

Funcionament: exemple pas a pas

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

fvar
inici




falgorisme
```

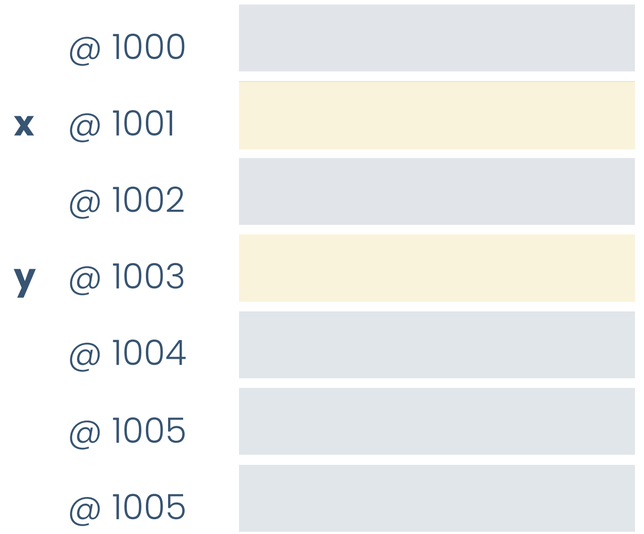


Memòria de 32 bits

Punters

Funcionament: exemple pas a pas

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

fvar
inici




falgorisme
```

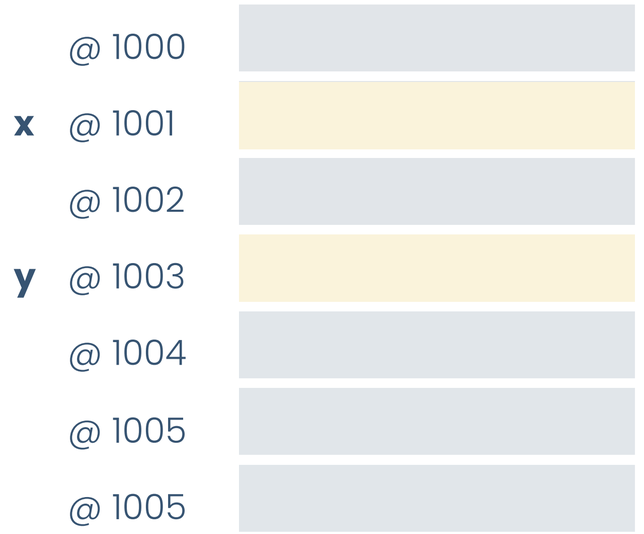


Memòria de 32 bits

Punters

Funcionament: exemple pas a pas

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









falgorisme
```

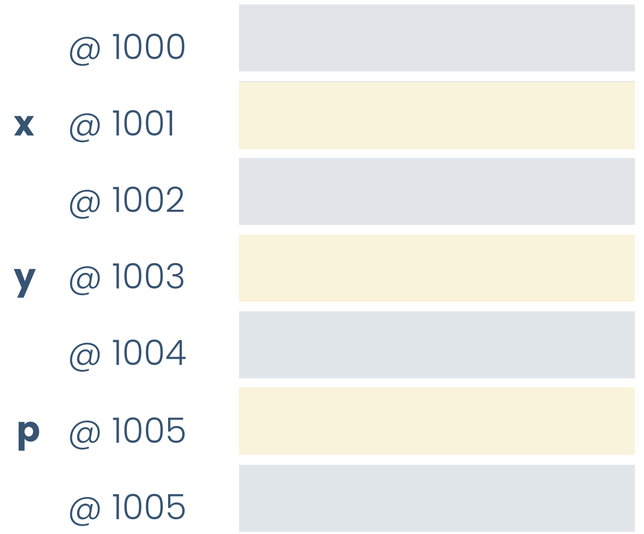


Memòria de 32 bits

Punters

Funcionament: exemple pas a pas

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









falgorisme
```

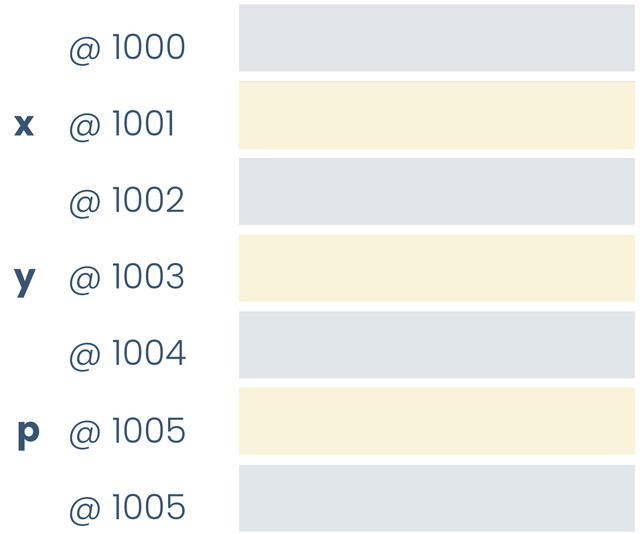


Memòria de 32 bits

Punters

Funcionament: exemple pas a pas

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




falgorisme
```

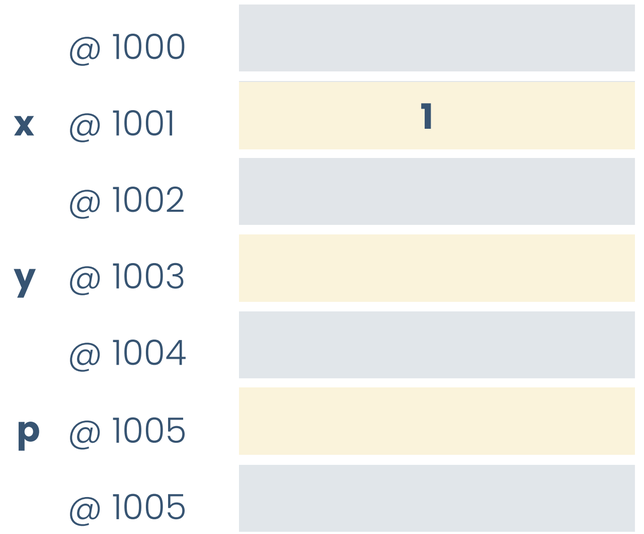


Memòria de 32 bits

Punters

Funcionament: exemple pas a pas

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




falgorisme
```

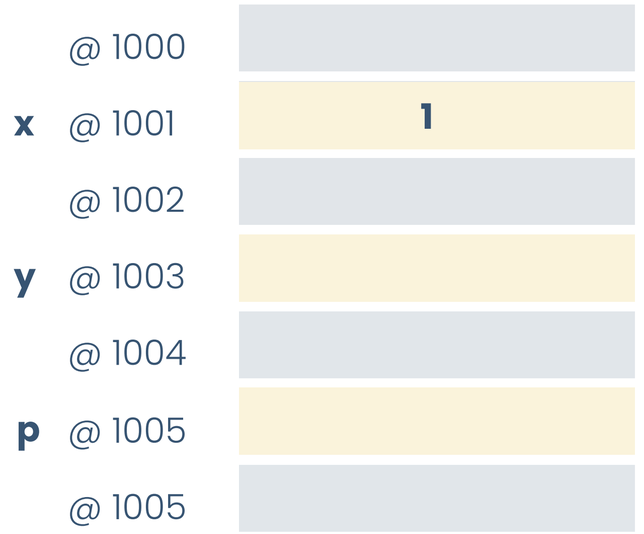


Memòria de 32 bits

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
```

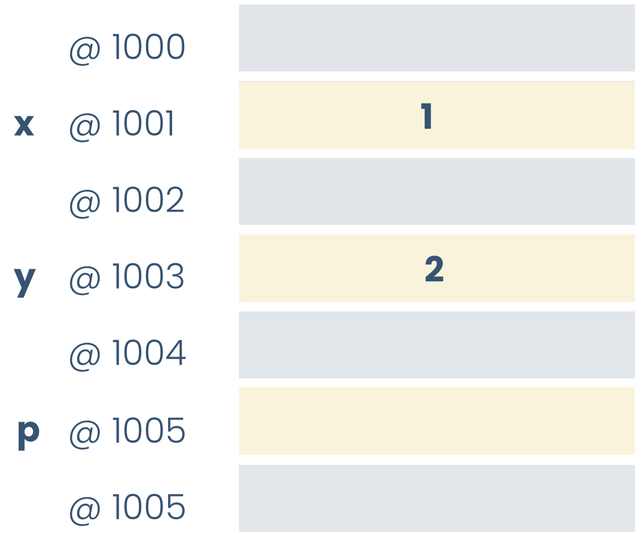


Memòria de 32 bits

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
```



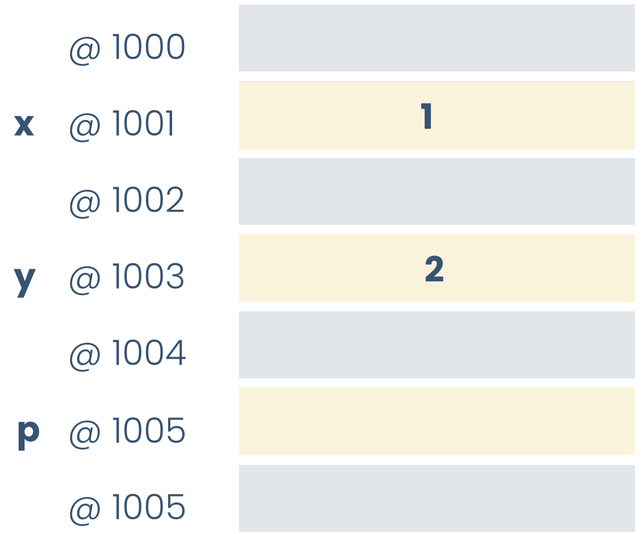
Memòria de 32 bits

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
```



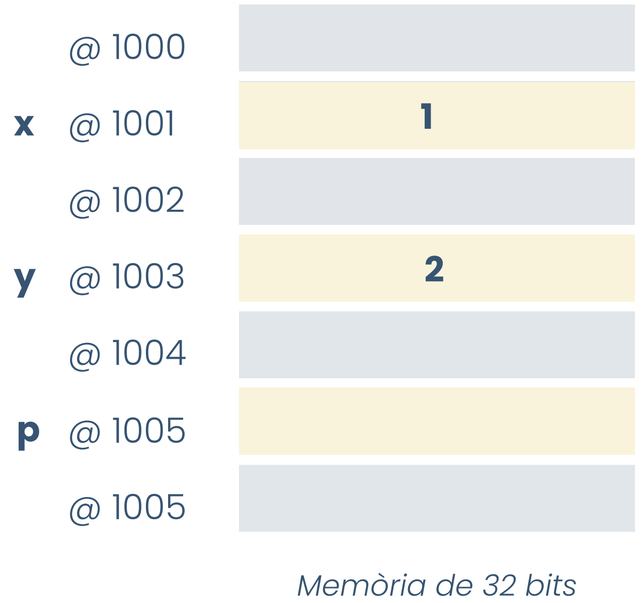
Memòria de 32 bits

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
```



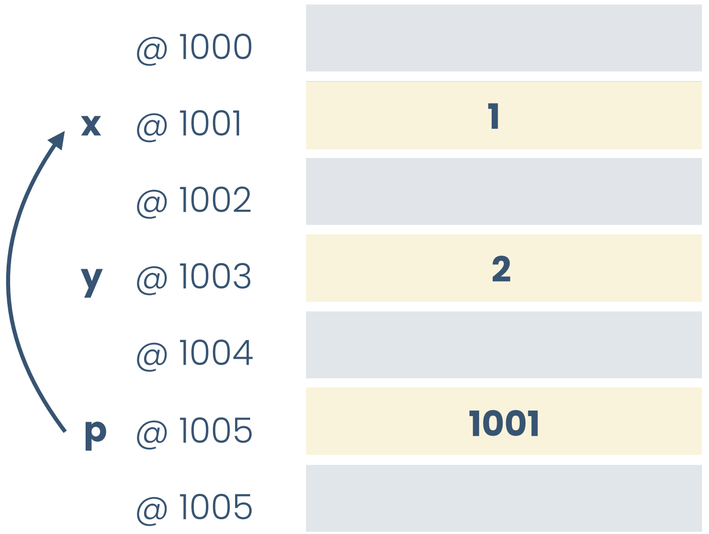
Memòria de 32 bits

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; // p apunta a x



falgorisme
```



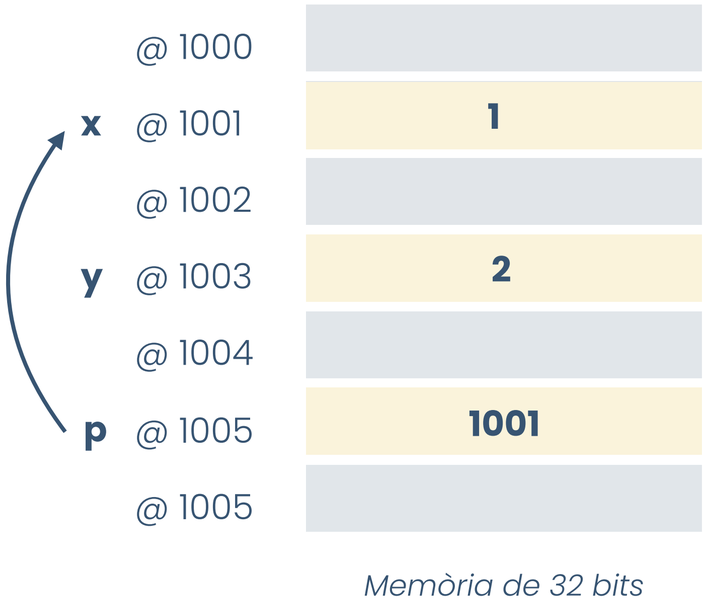
Memòria de 32 bits

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; // p apunta a x
  y := *p;


falgorisme
```



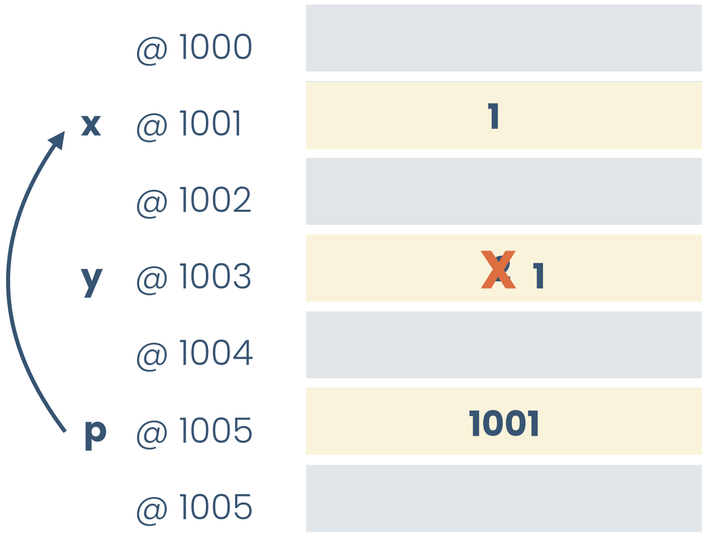
Memòria de 32 bits

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; // p apunta a x
   y := *p; // y val 1


falgorisme
```



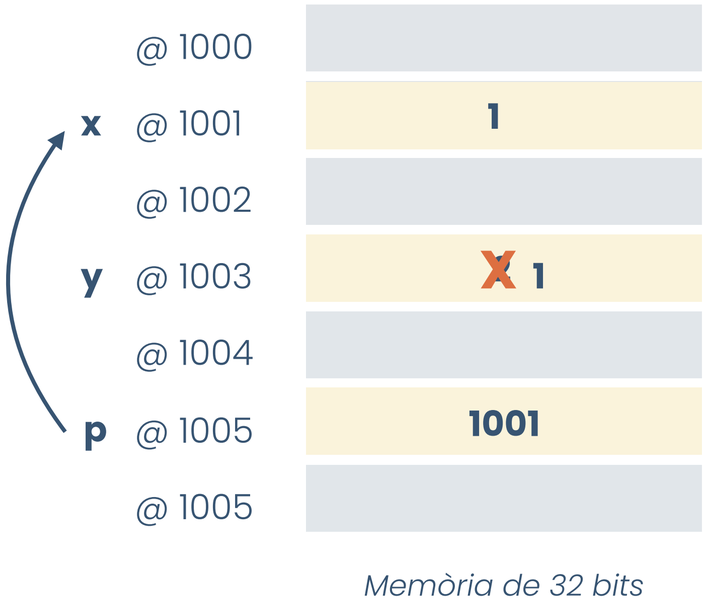
Memòria de 32 bits

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; // p apunta a x
   y := *p; // y val 1
   *p := 0;

falgorisme
```



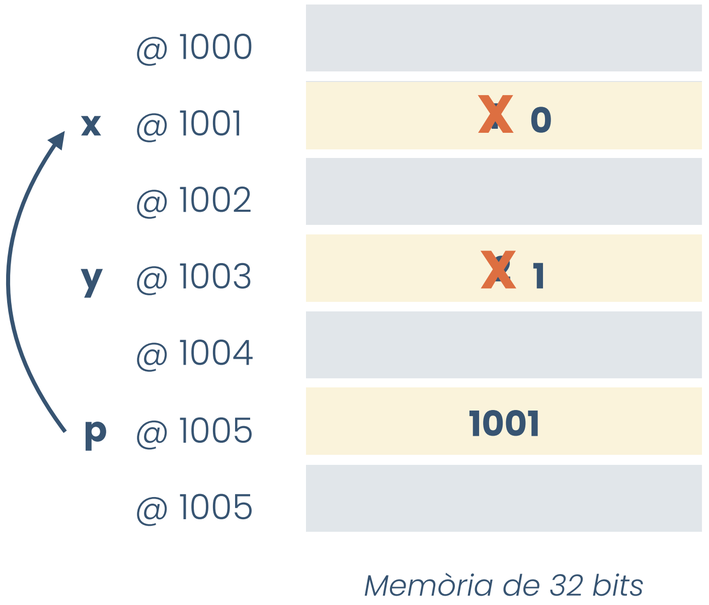
Memòria de 32 bits

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; // p apunta a x
   y := *p; // y val 1
   *p := 0; // x val 0

falgorisme
```



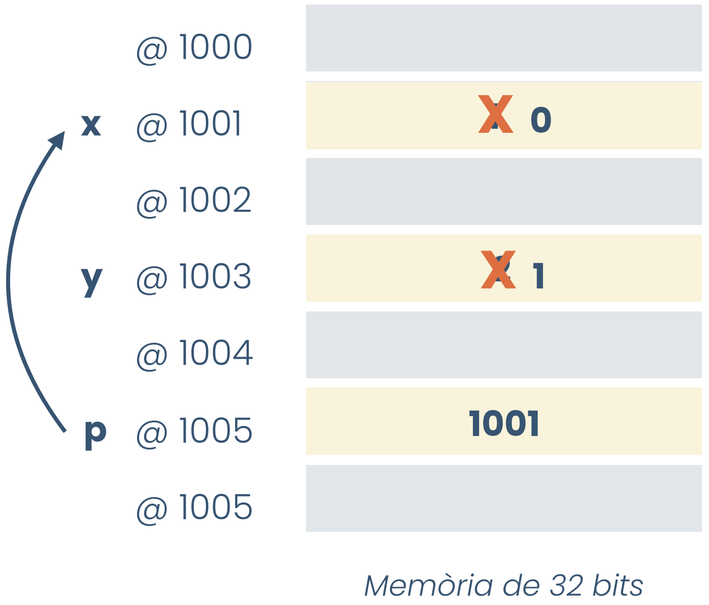
Memòria de 32 bits

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; // p apunta a x
   y := *p; // y val 1
   *p := 0; // x val 0

falgorisme
```



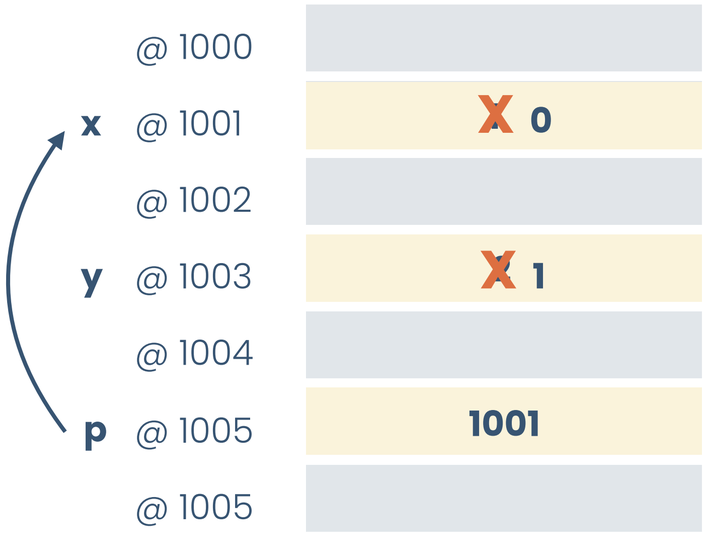
Memòria de 32 bits

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; // p apunta a x
  y := *p; // y val 1
  *p := 0; // x val 0
  p := &y;
falgorisme
```



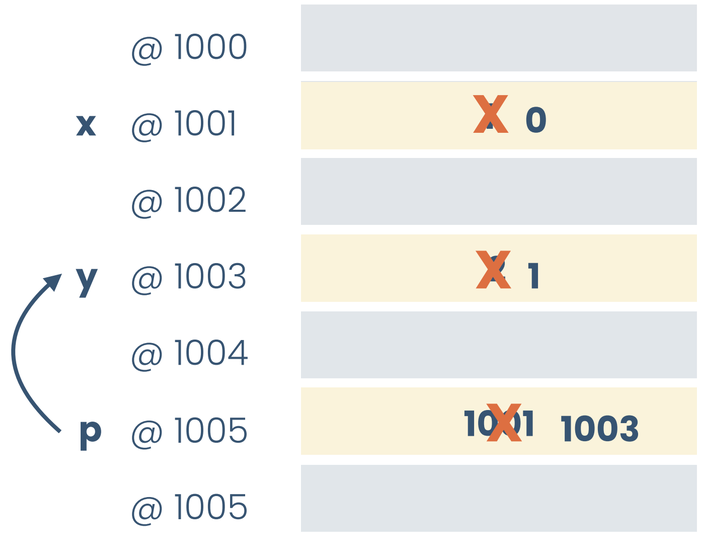
Memòria de 32 bits

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; // p apunta a x
  y := *p; // y val 1
  *p := 0; // x val 0
  p := &y; // p apunta a y
falgorisme
```



Memòria de 32 bits

Punters

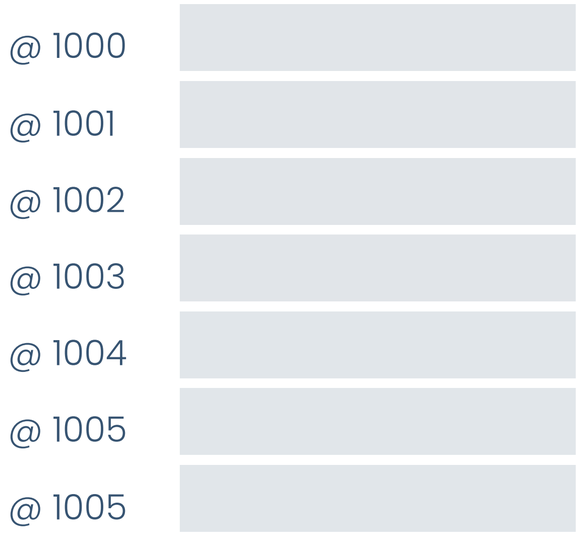
Operador "**&**": et dóna l'adreça de memòria d'un objecte
Operador "*****": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

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”
- Modifiqueu, a través del punter, el valor de “e”. Volem sumar-li 5 a “e”

```
algorisme exercici és
var



fvar
inici



falgorisme
```



Memòria de 32 bits

Punters

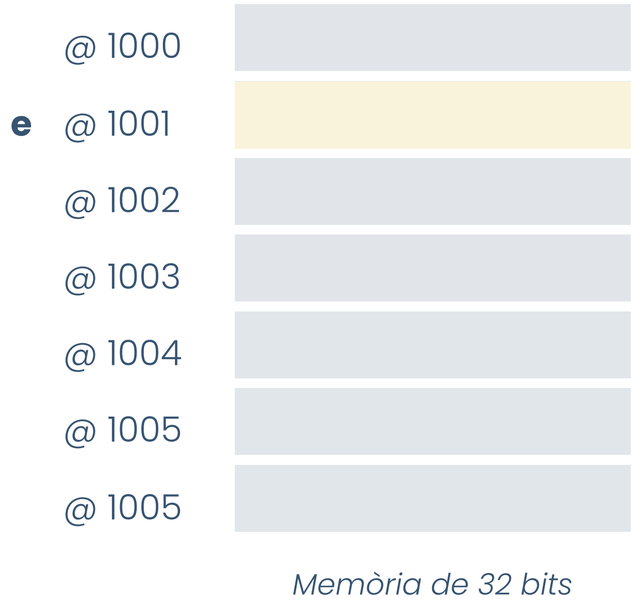
Operador "**&**": et dóna l'adreça de memòria d'un objecte
Operador "*****": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

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”
- Modifiqueu, a través del punter, el valor de “e”. Volem sumar-li 5 a “e”

```
algorisme exercici és
var
  e : enter;

fvar
inici



falgorisme
```



Memòria de 32 bits

Punters

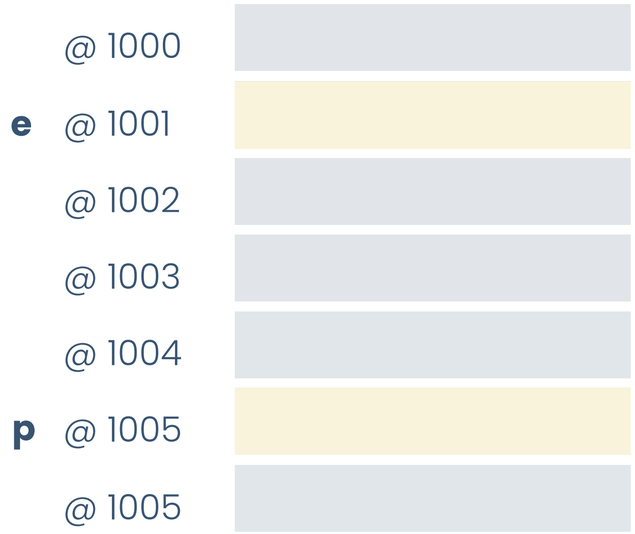
Operador "**&**": et dóna l'adreça de memòria d'un objecte
Operador "*****": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

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”
- Modifiqueu, 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
```



Memòria de 32 bits

Punters

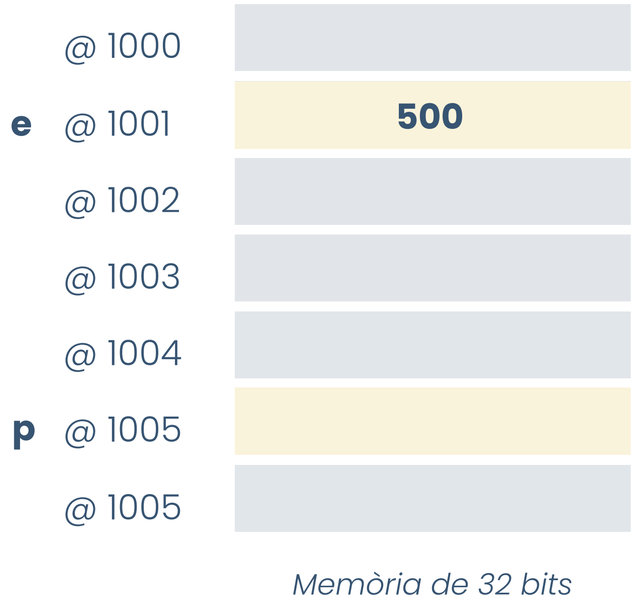
Operador "**&**": et dóna l'adreça de memòria d'un objecte
Operador "*****": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

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”
- Modifiqueu, 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
```



Memòria de 32 bits

Punters

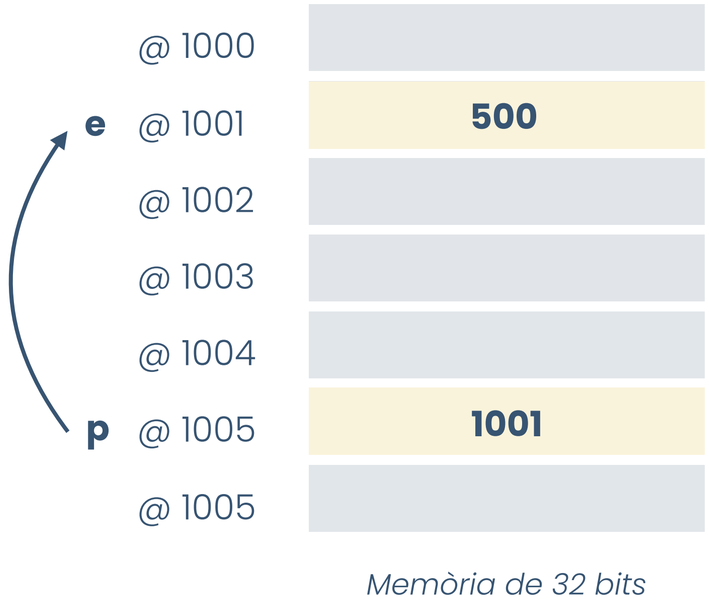
Operador "**&**": et dóna l'adreça de memòria d'un objecte
Operador "*****": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

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”
- Modifiqueu, 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
```



Memòria de 32 bits

Punters

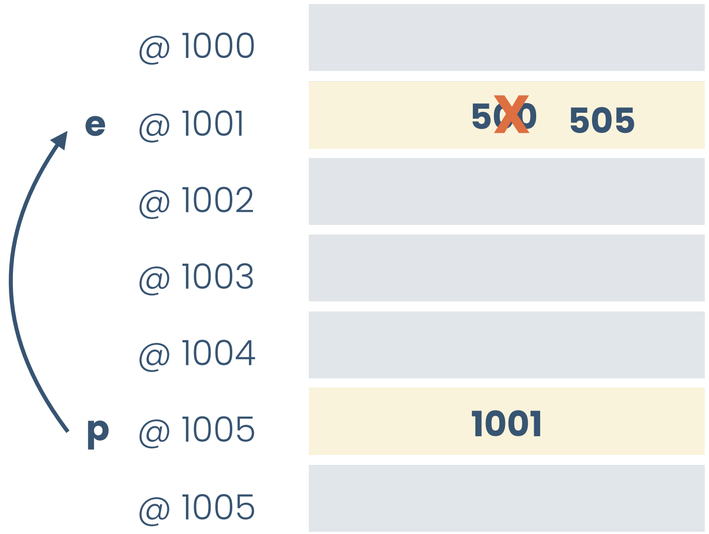
Operador "**&**": et dóna l'adreça de memòria d'un objecte
Operador "*****": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

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”
- Modifiqueu, 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
```



Memòria de 32 bits

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
```

Memòria de 32 bits

Punters

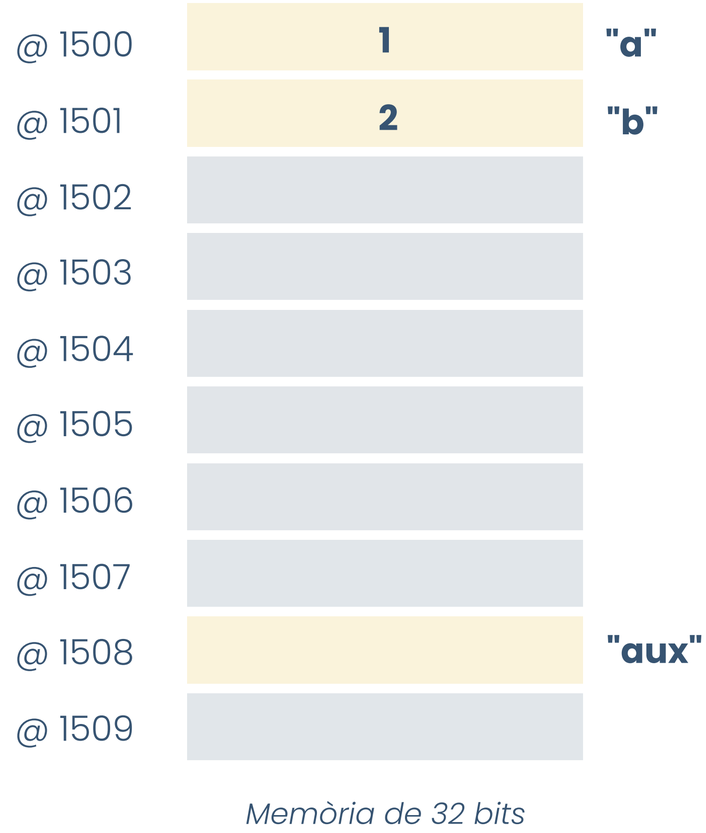
Operador "**&**": et dóna l'adreça de memòria d'un objecte
Operador "*****": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

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
```



Memòria de 32 bits

Punters

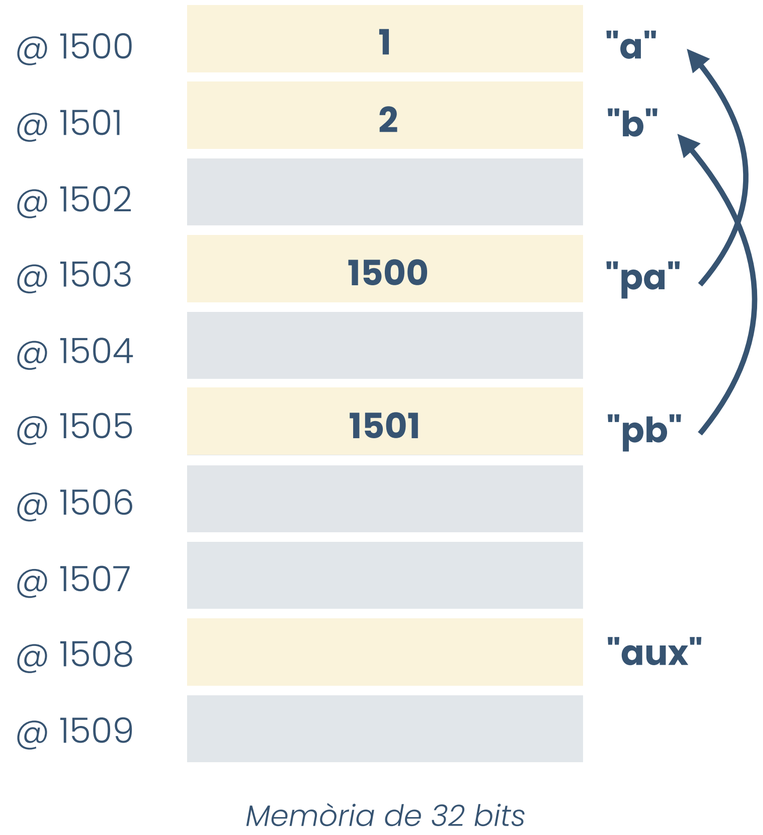
Operador "**&**": et dóna l'adreça de memòria d'un objecte
Operador "*****": aplicat a un apuntador, dona accés a l'objecte al qual s'apunta

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

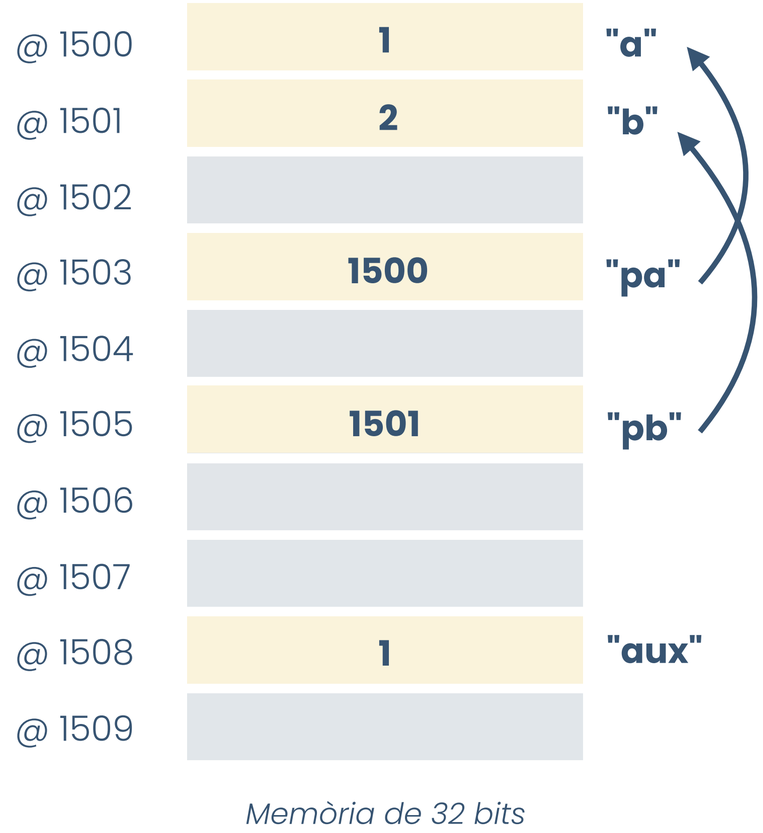
Exercici per practicar II

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)



Memòria de 32 bits

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

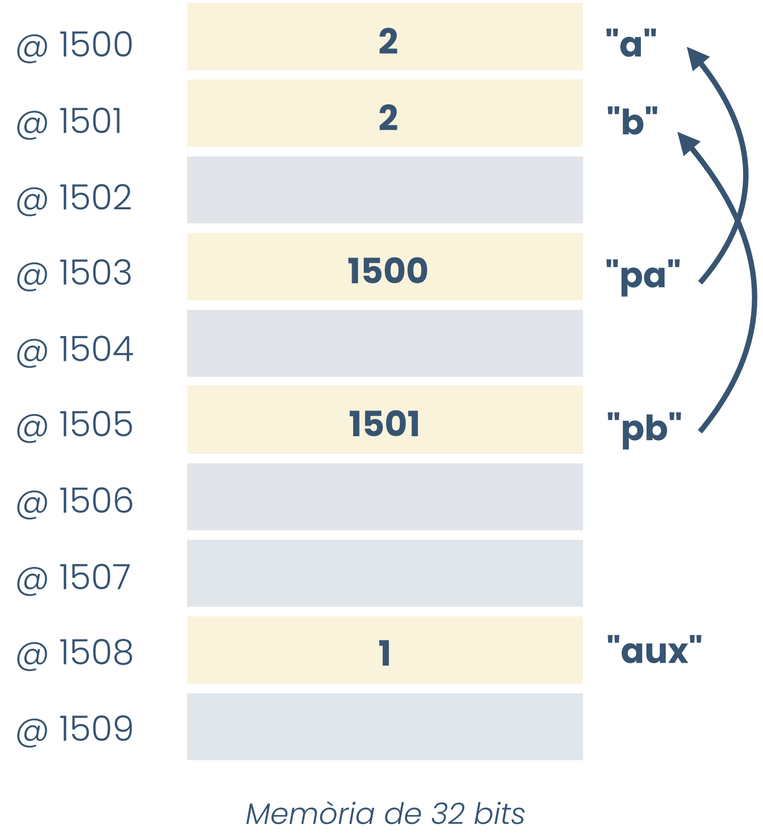
Escriu un algorisme que declari dues variables enteres **a** i **b** i en faci un intercanvi de valors fent servir punters.

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**)



Memòria de 32 bits

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.

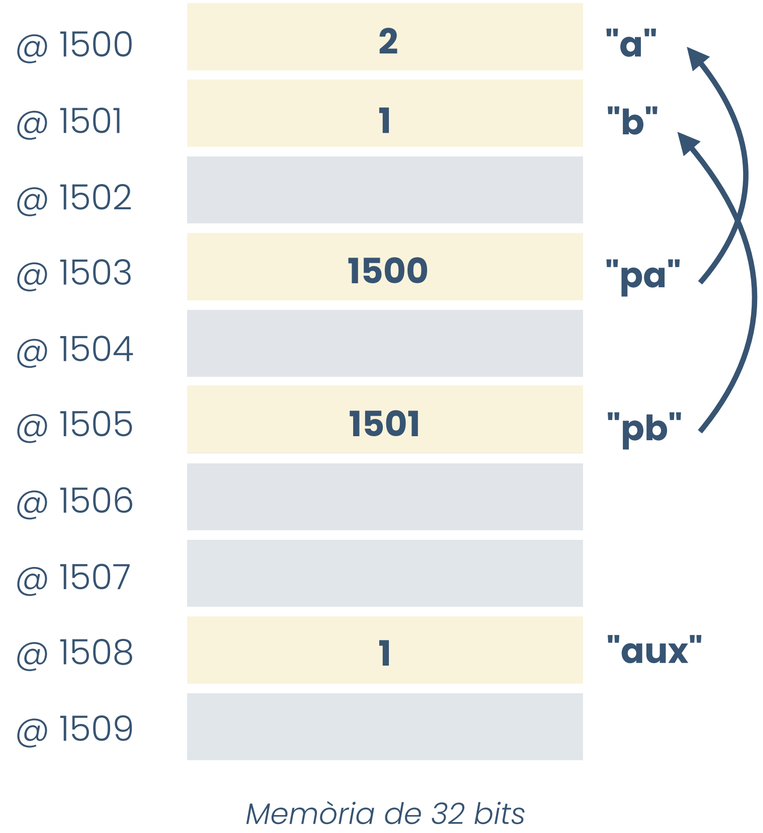
Pistes: necessitaràs dos punters i una variable auxiliar

```
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)



Memòria de 32 bits

PAS DE PARÀMETRES

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

Paràmetres **no** modificables

Qualsevol canvi que fem dins el procediment no tindrà efecte quan sortim del procediment

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

Pas per referència

Ho aprendrem a fer quan vegem punters

procediment es mantindran al sortir-ne

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

Cada llenguatge de programació gestiona el pas de paràmetres de manera diferent

En llenguatge C...

... es fa servir pas per valor

Això vol dir que mai podrem modificar els paràmetres?

No! Es fan servir adreces de memòria (anomenats punters o apuntadors) i llavors es pot accedir al contingut i modificar-lo

Utilitat dels punters

Pas de paràmetres per referència

Utilitat dels punters

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

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 ?

Utilitat dels punters

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

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 seran PUNTERS a enters.

```
...
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

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

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 seran PUNTERS a enters.

```
...
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




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;




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);



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 dígits.
- 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 dígits.
- 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 dígits.
- 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 dígits.
- 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
   dades: taula[MAX] de caràcters;
   maj, min, dig: enter;
fvar
inici
  comptar(dades, &maj, &min, &dig);
  escriure("Majúscules: ", maj);
  escriure("Minúscules: ", min);
  escriure("Dígits: ", dig);
falgorisme
```