

Ponteiros

Profa. Franciny Medeiros

O que são Ponteiros?

- » Ponteiros são variáveis que armazenam endereço de memória.
- » No C quando declaramos ponteiros nós informamos ao compilador para que tipo de variável vamos apontá-lo:
 - Os int's guardam inteiros;
 - Os float's guardam números reais;
 - Os Char's guardam caracteres.

Para que serve os ponteiros na programação?

- » Permitem a modificação de argumentos de funções: permitem que uma função altere valores de variáveis não globais e não locais a ela através da referência ao endereço de memória da variável passada como parâmetro para a função;
- » Permitem o uso de rotinas de alocação dinâmica de memória: alocação e desalocação de memória em tempo de execução conforme a necessidade do programa;
- » Aumento de eficiência em determinadas rotinas.

Como declarar um ponteiro?

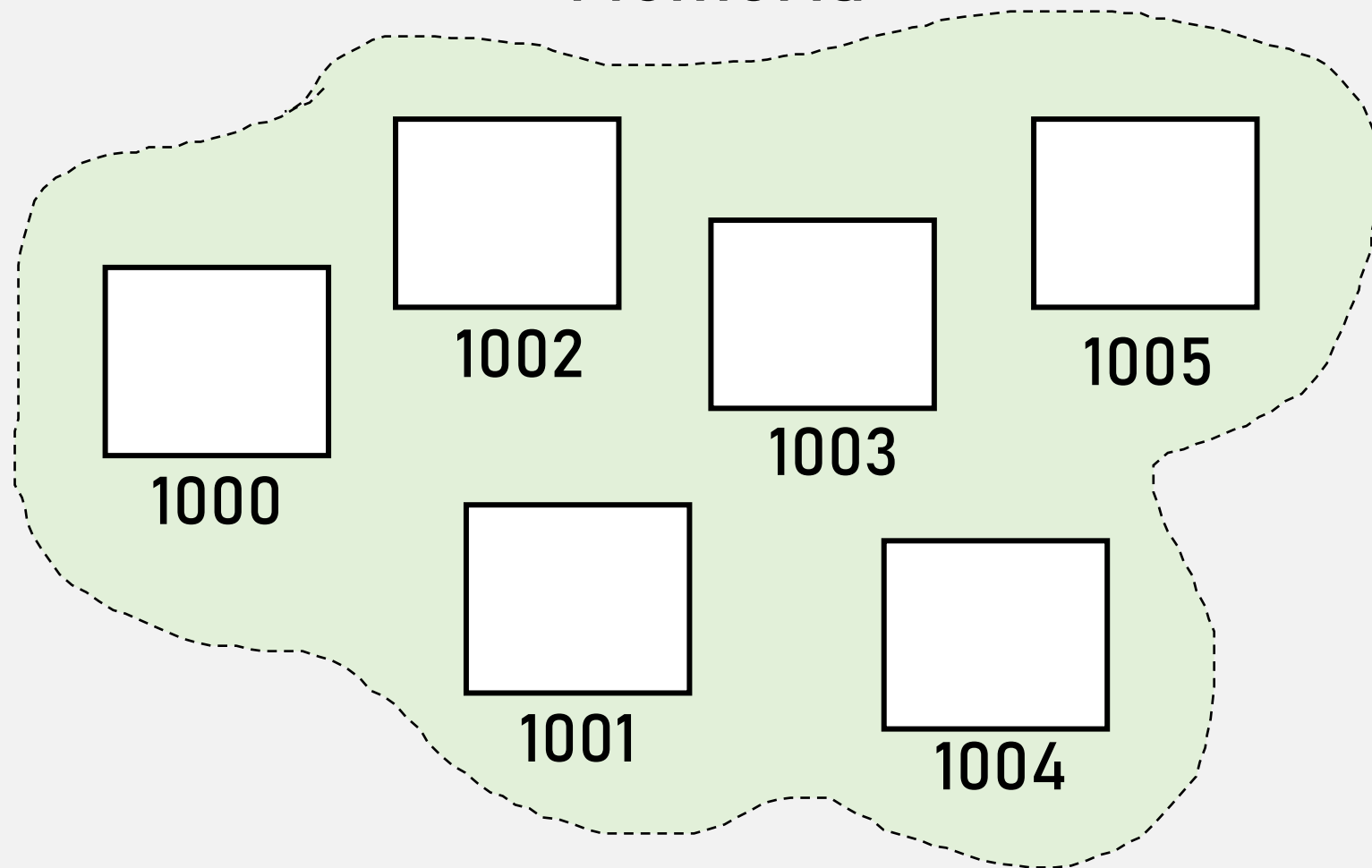
`tipo_do_ponteiro *nome_da_variável;`

- » `tipo_do_ponteiro` é o tipo de variável apontada pela variável ponteiro.
- » `int *P`, por exemplo, é um ponteiro de inteiros, ou seja, `P` apontará para uma região de memória que armazena um valor inteiro.

Exemplo

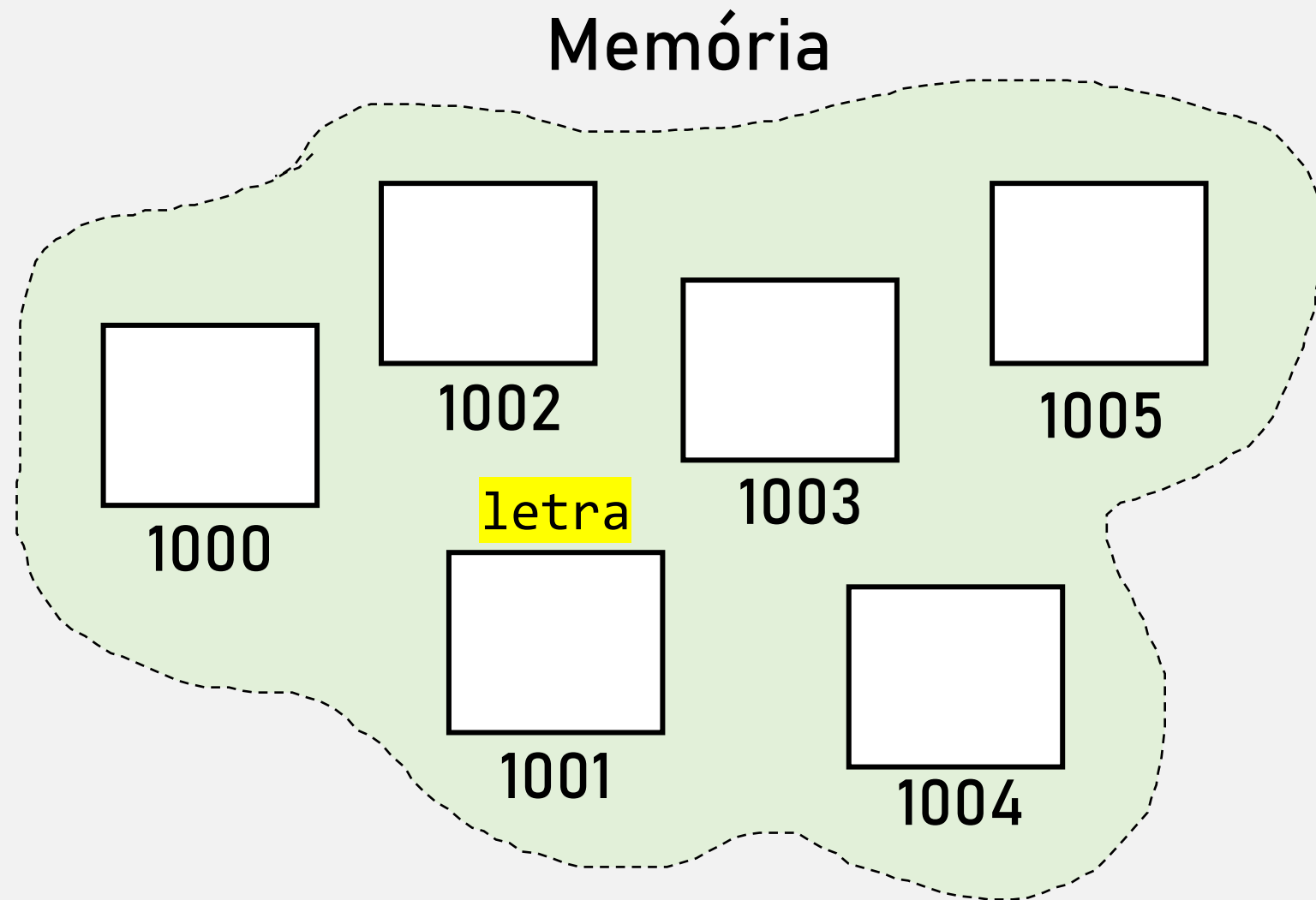
char letra;

Memória



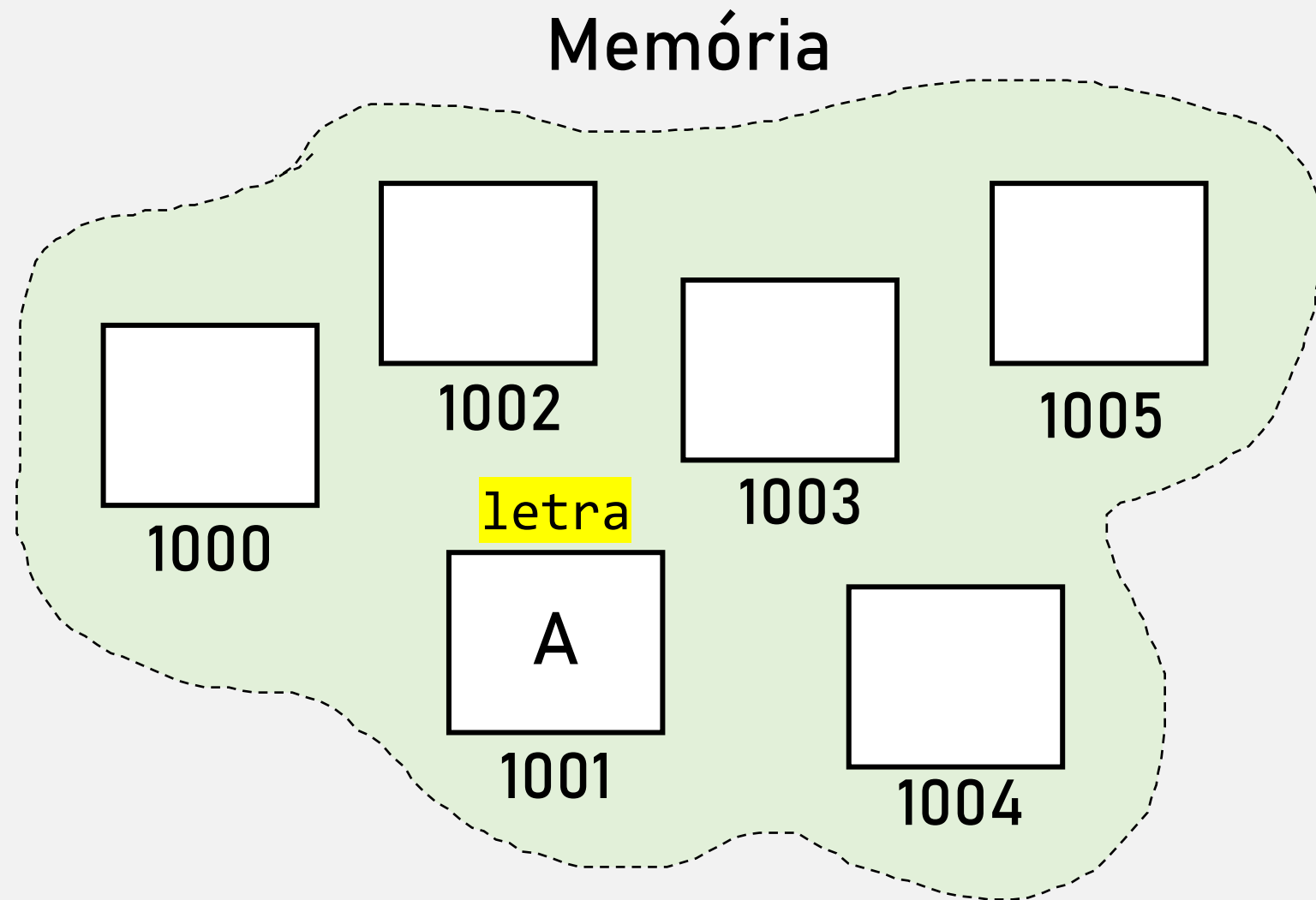
Exemplo

```
char letra;  
letra = 'A';
```



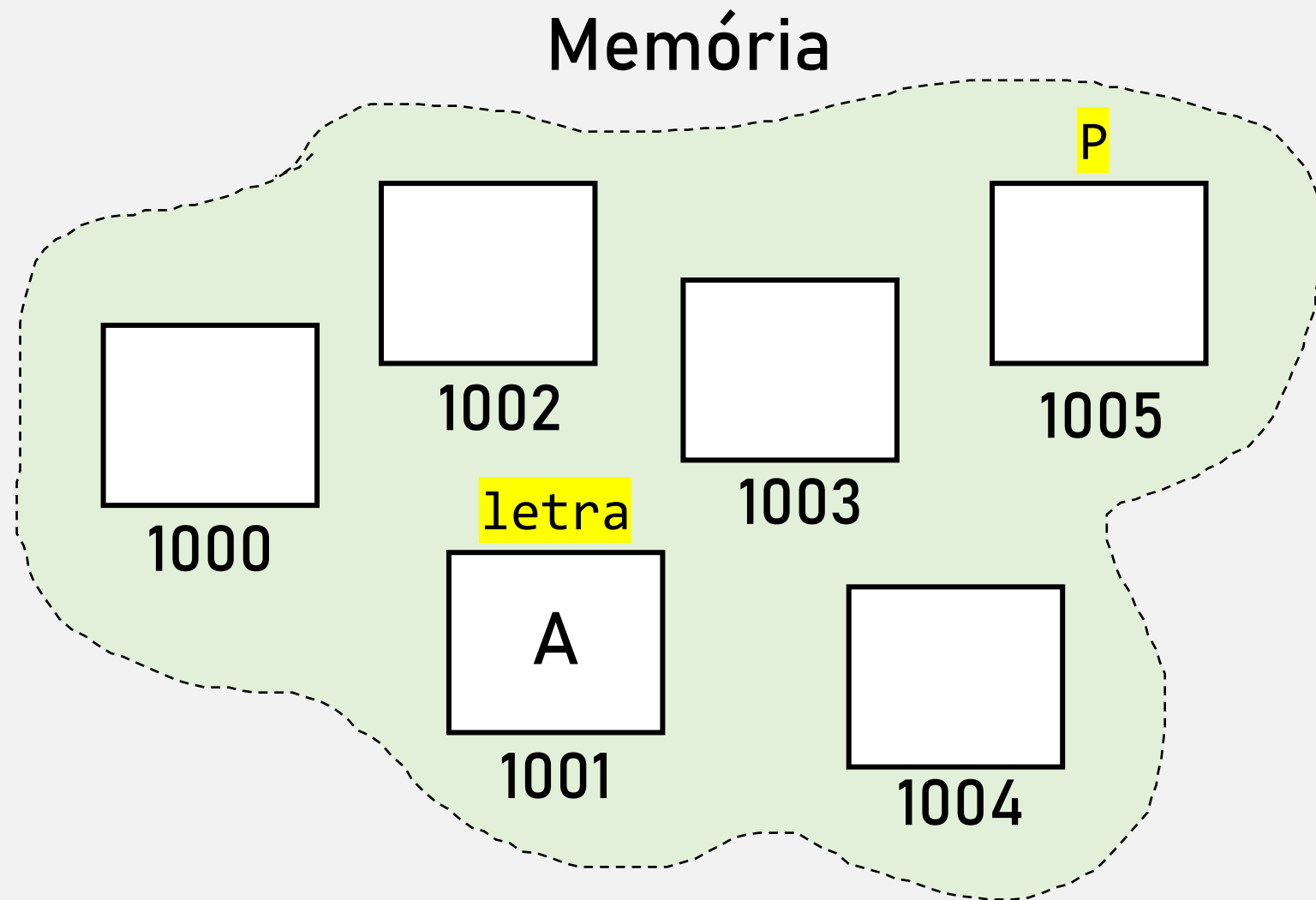
Exemplo

```
char letra;  
letra = 'A';  
char *P;
```



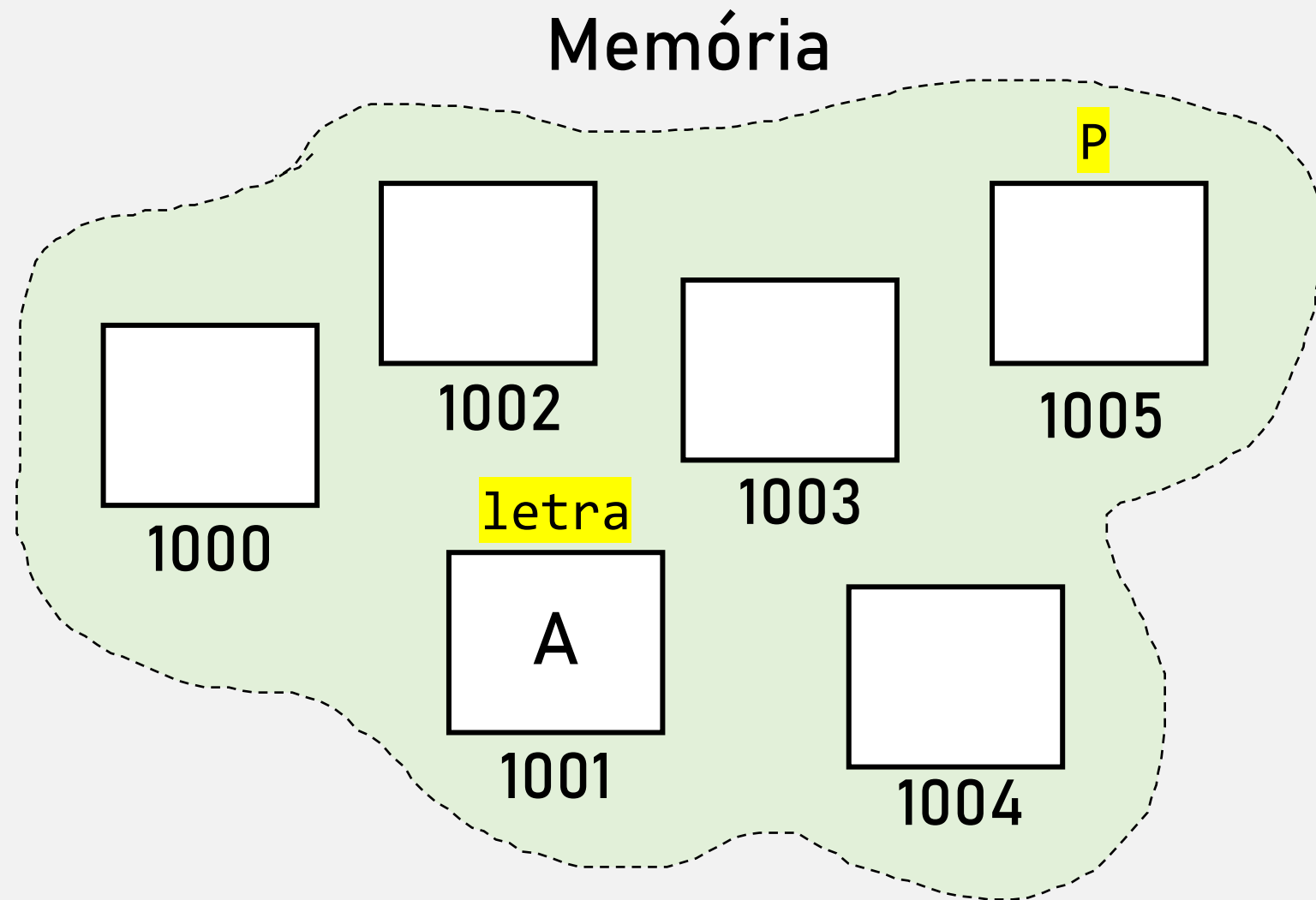
Exemplo

```
char letra;  
letra = 'A';  
char *P;
```



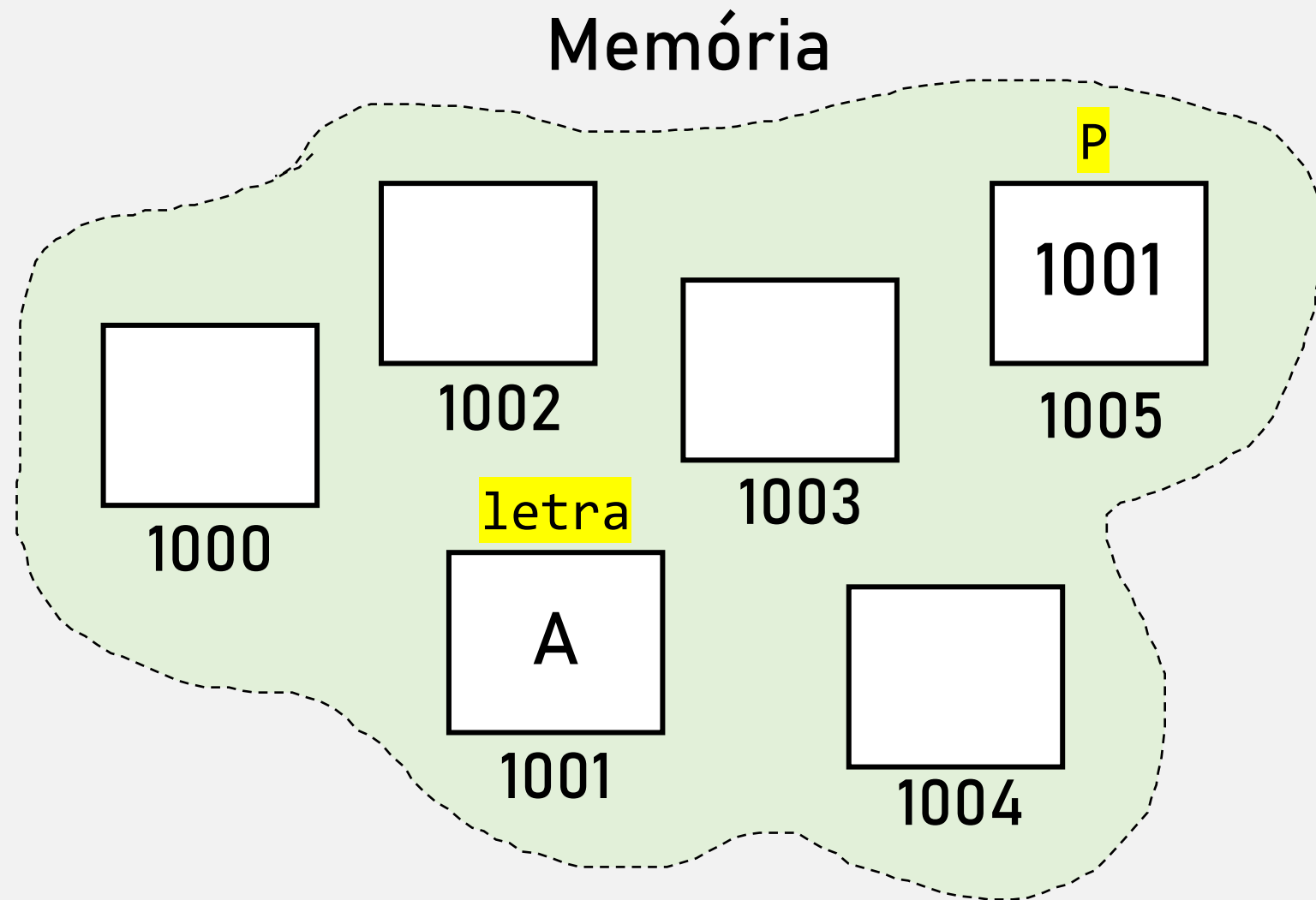
Exemplo

```
char letra;  
letra = 'A';  
char *P;  
P = &letra;
```



Exemplo

```
char letra;  
letra = 'A';  
char *P;  
P = &letra;
```



O ponteiro armazena o endereço de memória de uma variável!

Operadores para ponteiros

» Existem dois operadores especiais para trabalhar com ponteiros. São eles:

* - conteúdo do endereço apontado por

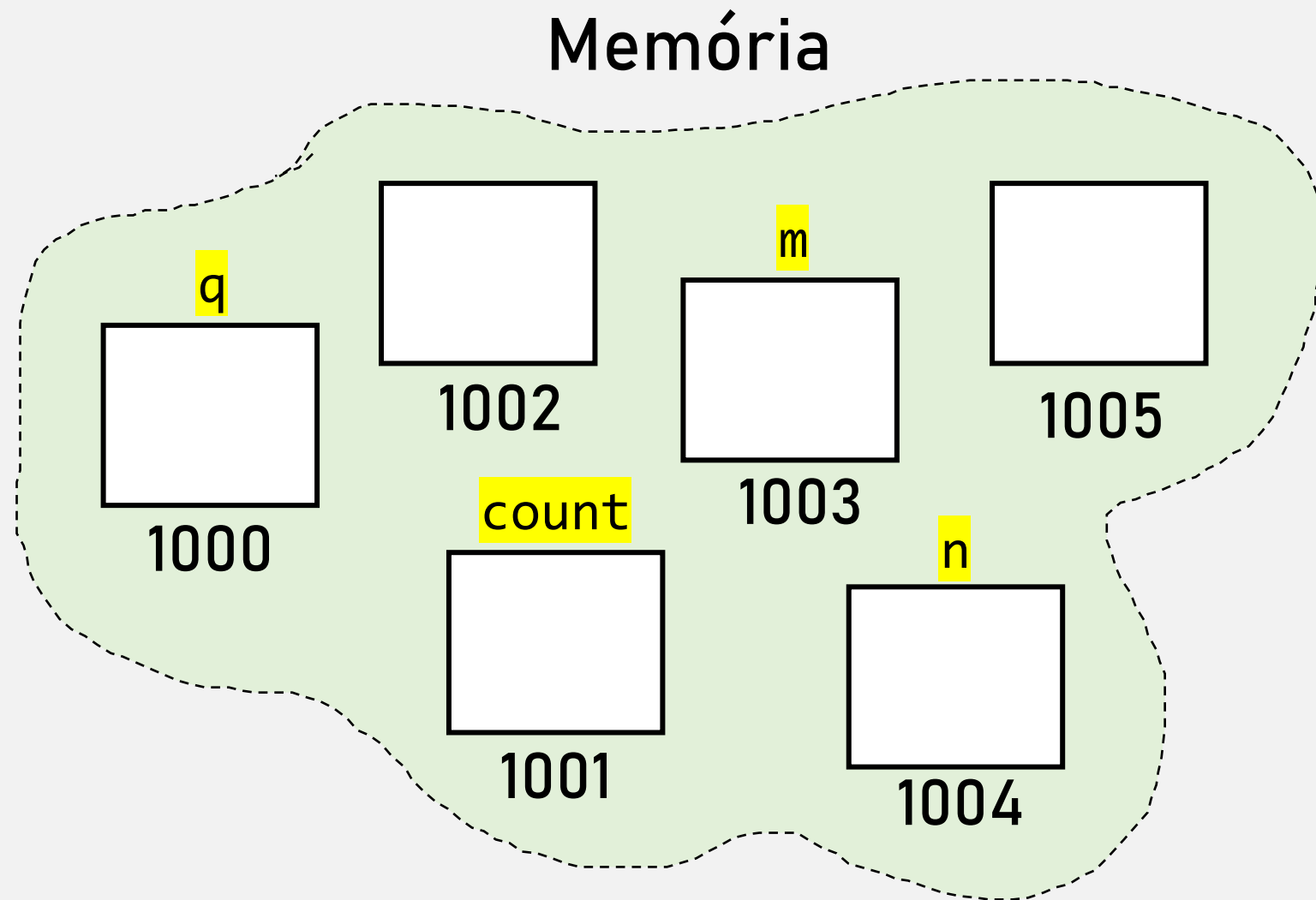
& - endereço de

Exemplo

```
1  main(){
2      int v1, v2, *p, *q;
3      v1 = 3;
4      v2 = 12;
5      p = &v1; //p recebe o endereço de memória da variável v1
6      q = p;    //copia o endereço guardado em p para q
7      *q = 44; //altera o valor armazenado no endereço apontado por q
8  }
```

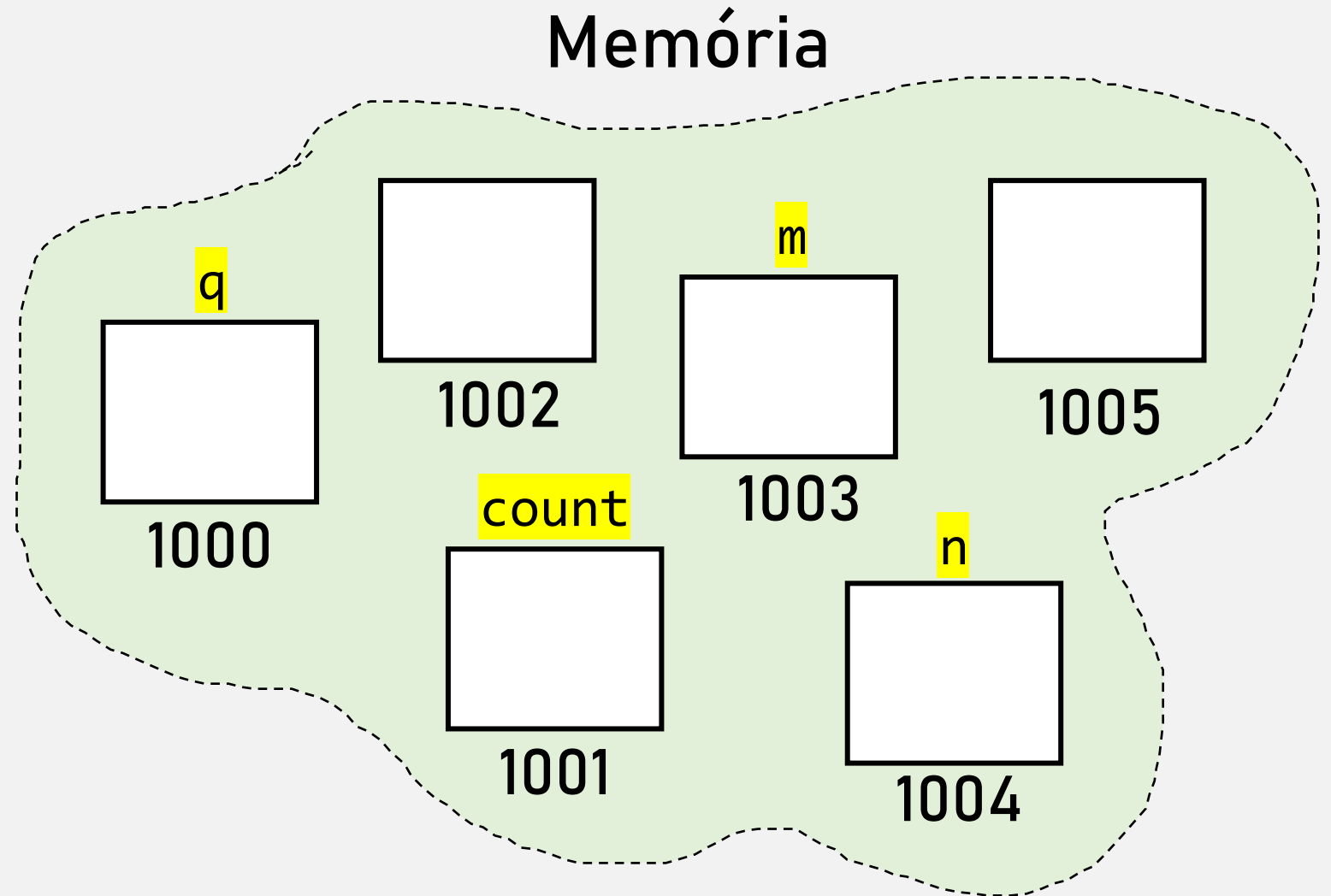
Exemplo

```
int count;  
int q;  
int *m;  
int *n;
```



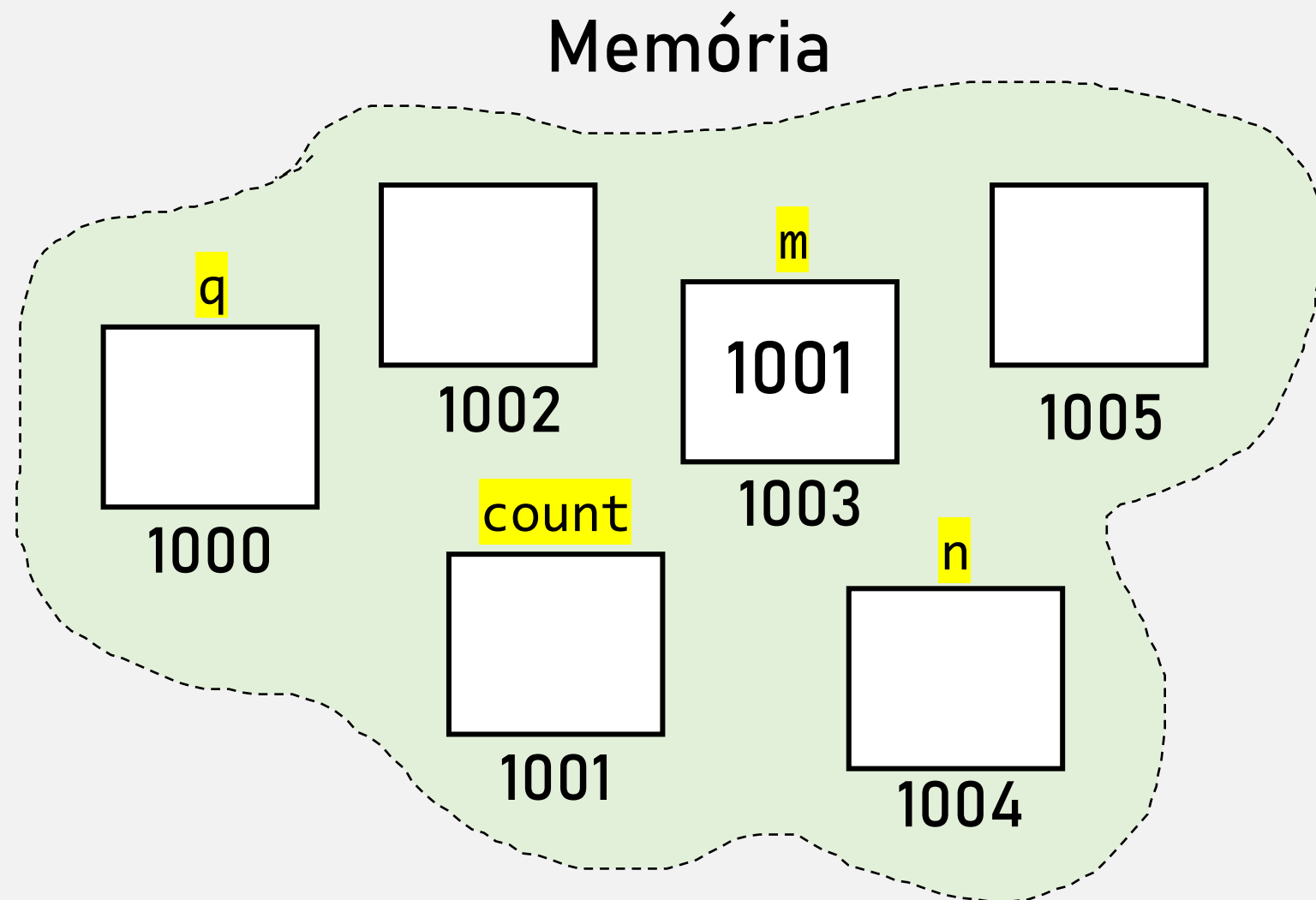
Exemplo

```
int count;  
int q;  
int *m;  
int *n;  
m = &count;
```



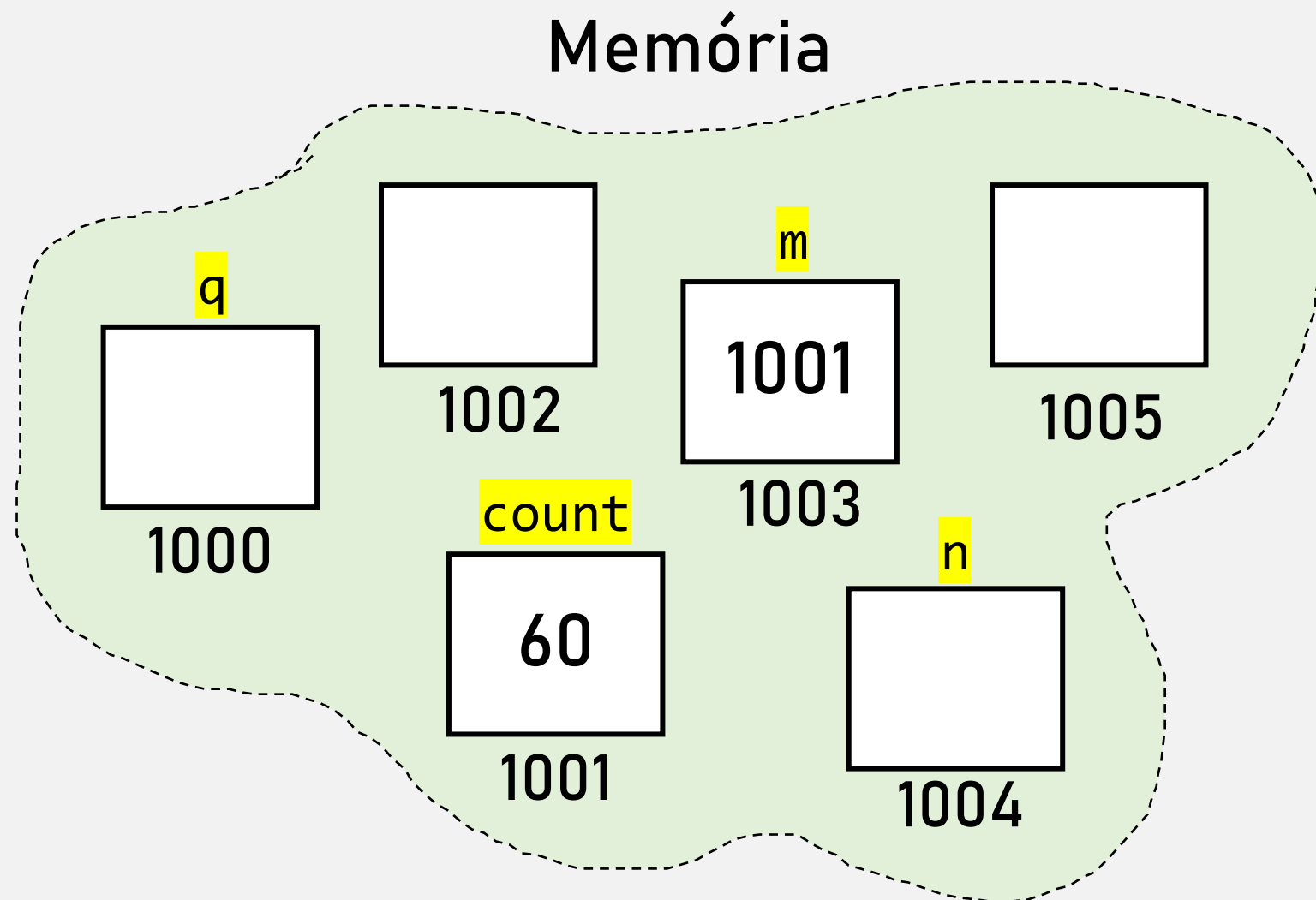
Exemplo

```
int count;  
int q;  
int *m;  
int *n;  
m = &count;  
count = 60;
```



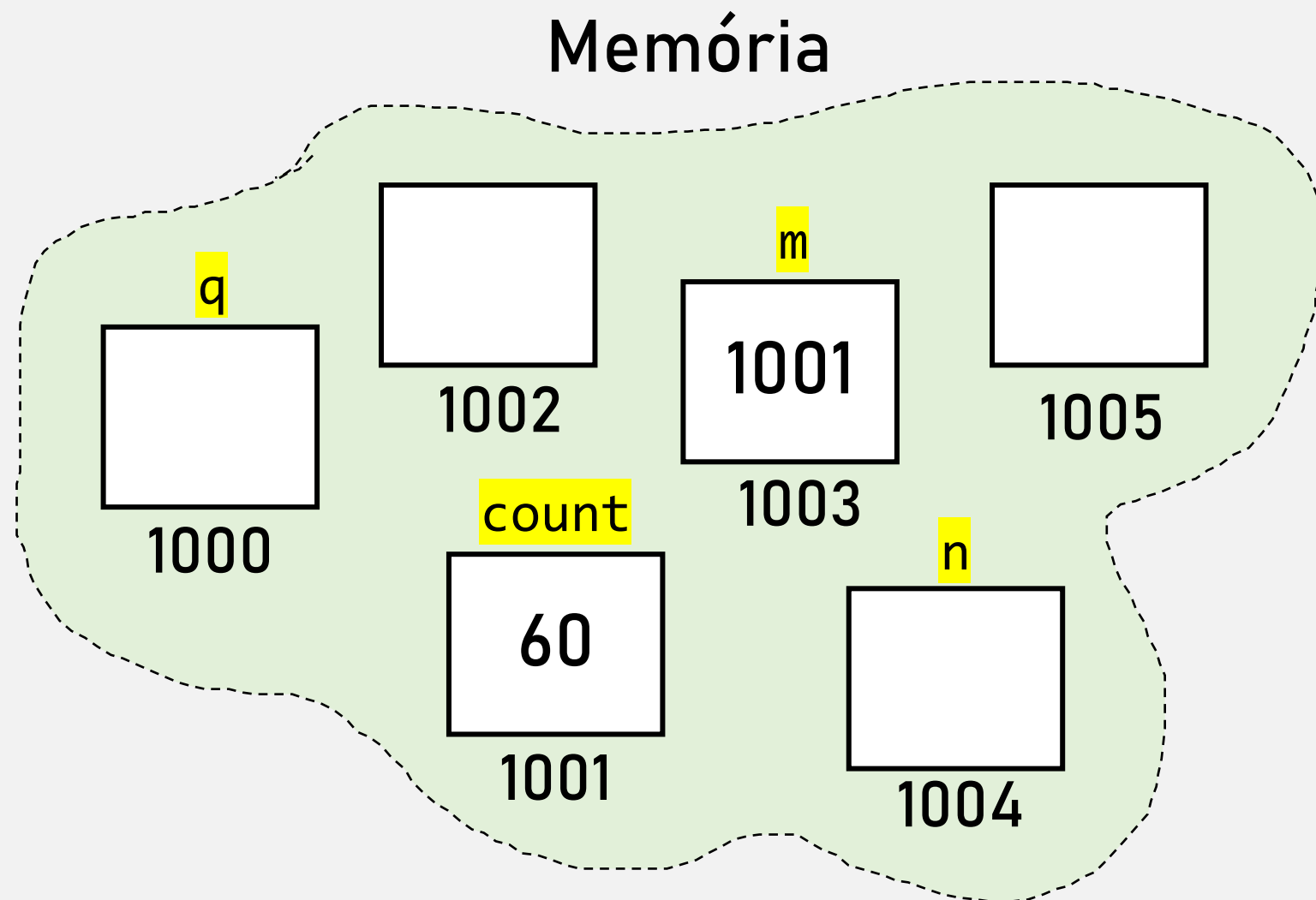
Exemplo

```
int count;  
int q;  
int *m;  
int *n;  
m = &count;  
count = 60;
```



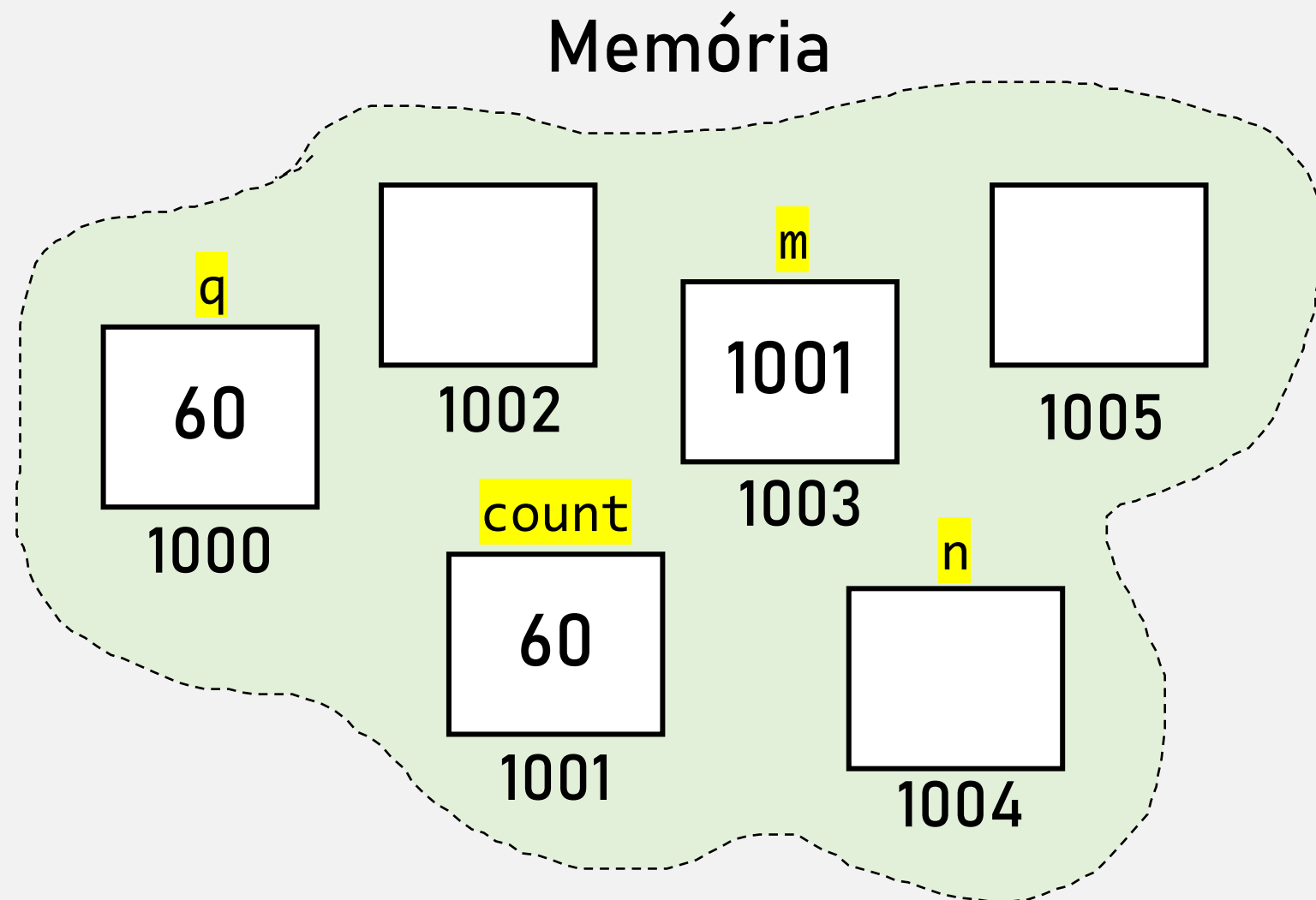
Exemplo

```
int count;  
int q;  
int *m;  
int *n;  
m = &count;  
count = 60;  
q = *m;
```



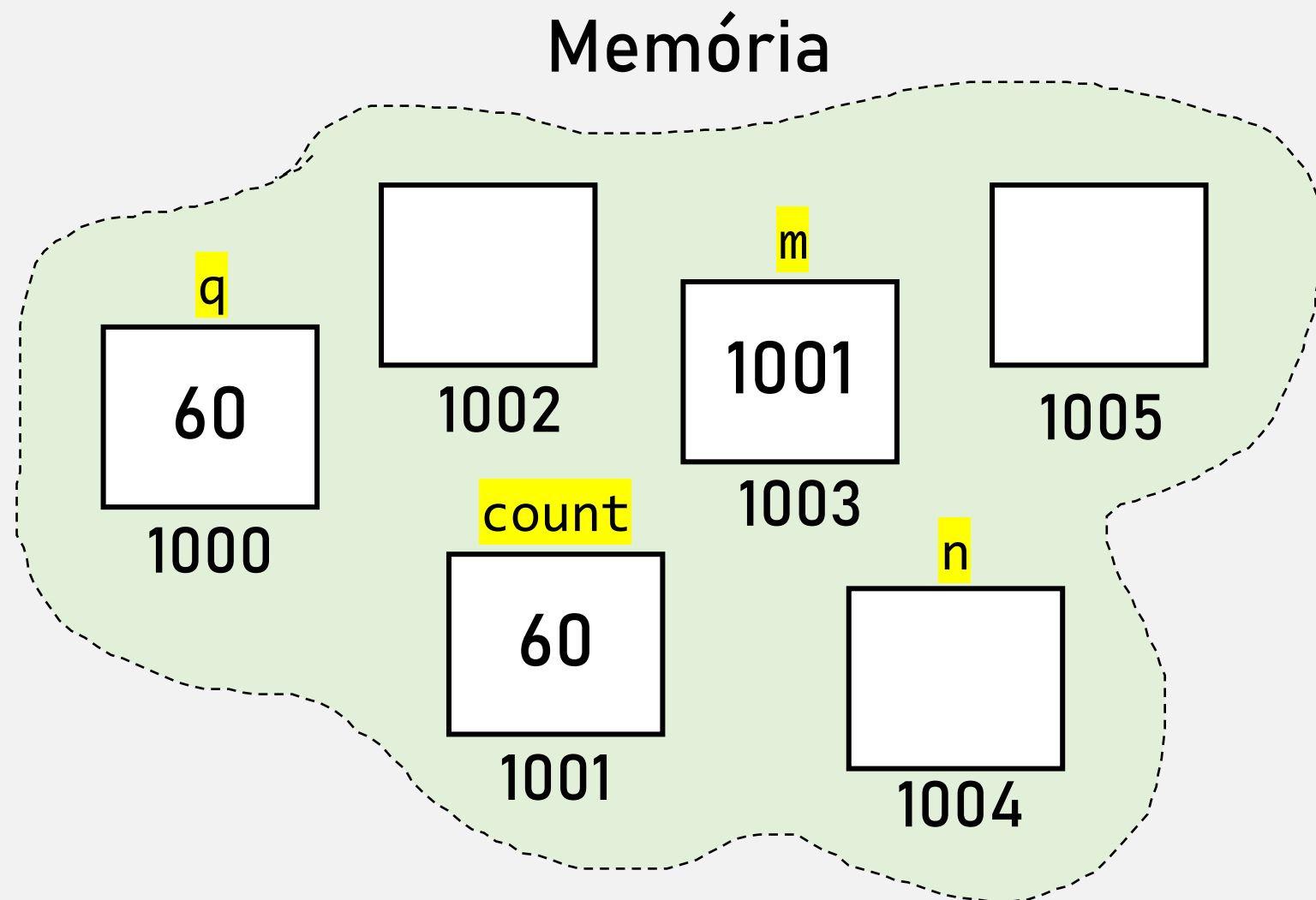
Exemplo

```
int count;  
int q;  
int *m;  
int *n;  
m = &count;  
count = 60;  
q = *m;
```



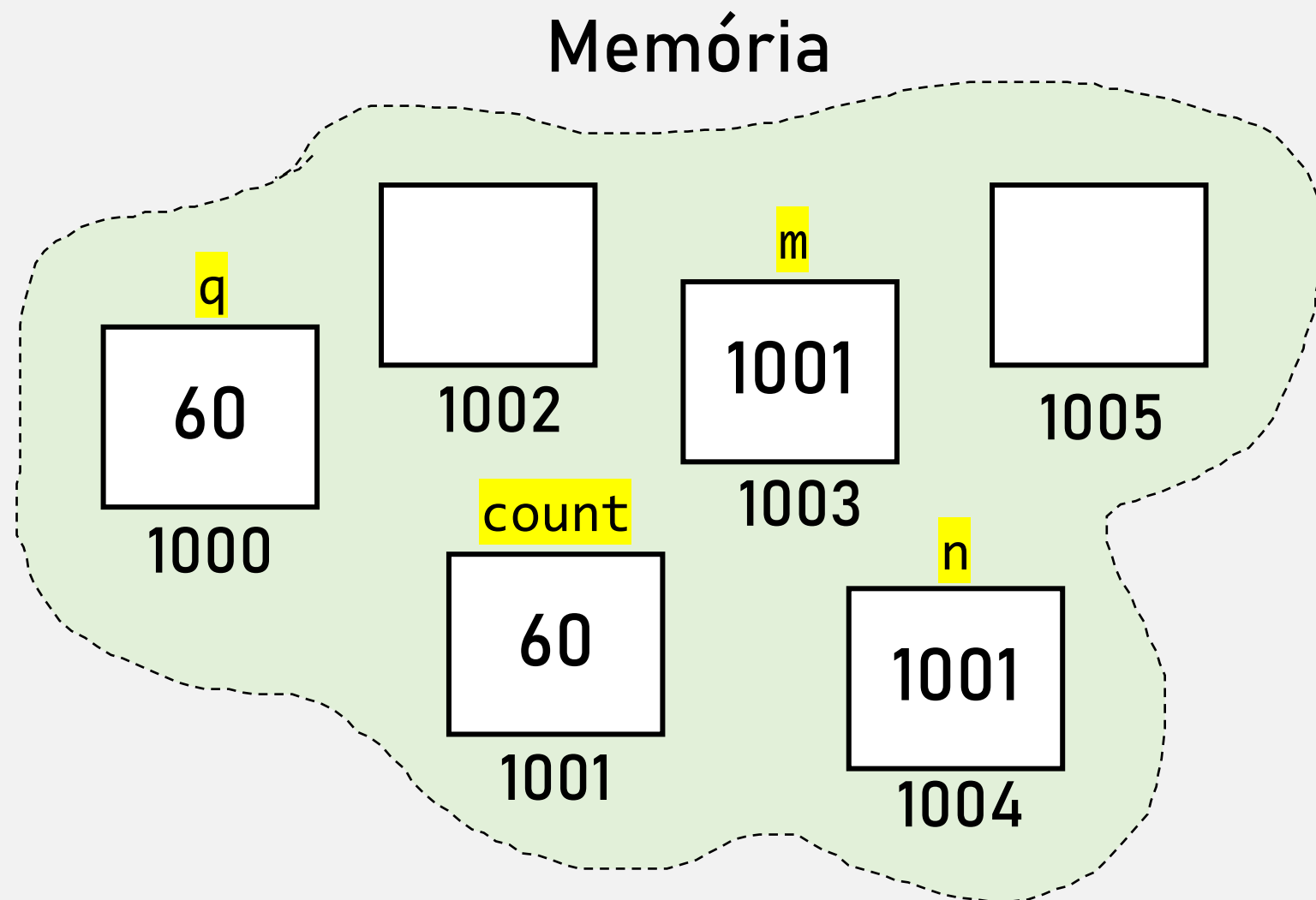
Exemplo

```
int count;  
int q;  
int *m;  
int *n;  
m = &count;  
count = 60;  
q = *m;  
n = m;
```



Exemplo

```
int count;  
int q;  
int *m;  
int *n;  
m = &count;  
count = 60;  
q = *m;  
n = m;
```



Chamada de funções

- » Existem duas maneiras de enviar valores na chamada de uma função, conhecidas como:
 - » passagem de parâmetro por referência;
 - » passagem de parâmetro por valor.

Chamada de funções

- » Na chamada de função por valor, o conteúdo de uma variável é copiado para o parâmetro da função.
- » Desta maneira, qualquer modificação que ocorra dentro da função chamada não altera o valor original das variáveis passadas nos argumentos.
- » Exemplo de código

Chamada de funções

- » Na chamada de função por referência, é enviado o endereço da memória onde o argumento está armazenado.
- » Sendo assim, a função deve estar preparada para receber um endereço e seus parâmetros serão do tipo **ponteiro**. Dentro da função este endereço será utilizado para alterar o valor do argumento, uma vez que acessa a posição de memória onde ele está.