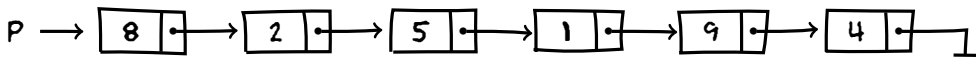
Imprimindo os elementos das posições ímpares

Imagine que o ponteiro `p` aponta para uma lista encadeada



A tarefa é

→ *Imprimir os elementos das posições ímpares da lista*

Bom, a ideia aqui é utlizar a passagem de parâmetro outra vez.

Quer dizer, as chamadas recebem um parâmetro indicando se elas estão em uma posição par ou ímpar.

E quando elas fazem a sua chamada recursiva, elas invertem o valor do parâmetro.

Daí, as chamadas que estiverem em uma posição ímpar imprimem o valor do seu elemento.

A coisa fica assim

```
func imprimePI-Rec (q, pd)
      Se ( q == Nulo )   Retorna                  // já acabei ...
      Sse ( q->num == k )
            Imprime (q->num)
      imprimePI-Rec (q->prox, 1 - pd)             // inverte o valor de pd
      Retorna
```

E a função `main()` fica assim

```
func main ()
      imprimePI-Rec (p, 1)
```

Mas essa não é a única maneira de fazer as coisas.

Quer dizer, a gente também pode percorrer a lista pulando de 2 em 2 — (*i.e., passando apenas pelas posições ímpares*)

```
imprimePI2-Rec ((q->prox)->prox)
```

Aqui, é preciso ter cuidado para não tentar dar o salto de tamanho 2 na última posição.

Porque na última posição

```
(q->prox)->prox   ≡   (Nulo)->prox
```

Só que o `Nulo` não tem um campo `prox` — (*o que dá um erro de execução ...*)

A coisa fica assim

```
func imprimePI2-Rec (q)
      Se ( q == Nulo )   Retorna                  // já acabei ...
      Imprime (q->num)
      Se ( q->prox != Nulo )                      // eu não sou o último?
            imprimePI-Rec ((q->prox)->prox)
      Retorna
```

◇