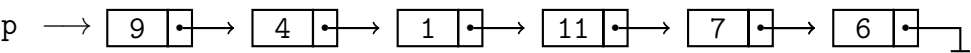


Imagine que o ponteiro `p` aponta para uma lista encadeada



A tarefa é

→ *Imprimir os elementos das posições ímpares da lista*

Bom, uma lista encadeada não tem posições numeradas — (*como os vetores*)

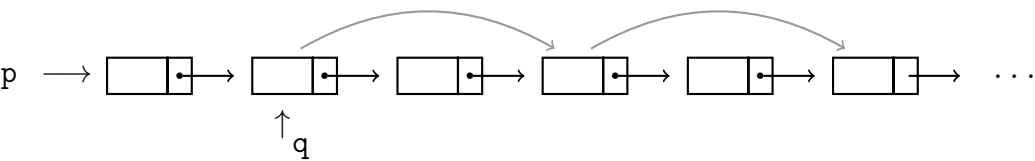
Então para realizar a tarefa, a gente pode ir contando as posições a medida que a gente vai passando por elas.

A coisa fica assim

```
q = p;    pos = 0                                // a 1o posição é a posição 0
Enquanto ( q != Nulo )                          // percorre a lista
    Se (pos é impar)
        Imprime(q->num)
    q = q->prox;    pos++                        // anda o ponteiro e a posição
```

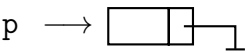
Mas, essa não é a única maneira de fazer as coisas.

Quer dizer, outra ideia é começar na posição 1 e ir pulando de 2 em 2

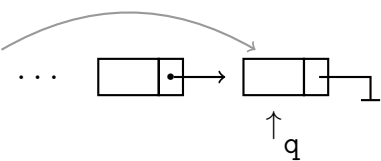


Mas aqui é preciso ter algum cuidado.

Porque pode não haver ninguém na posição 1



E porque não é possível dar um pulo de tamanho 2 a partir da última posição



Porque?

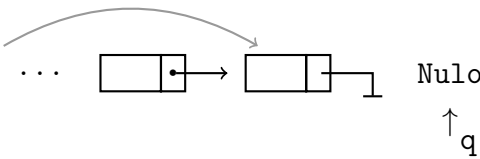
Ora, porque na prática o pula de tamanho 2 são dois pulos de tamanho 1

```
q = q->prox
q = q->prox
```

o que também pode ser escrito assim

```
q = (q->prox)->prox
```

Daí, quando a gente está na última posição, o primeiro pulo leva a gente para o **Nulo**



E não se pode dar um pulo a partir do **Nulo** — (*porque o Nulo não é uma caixinha que tem um ponteiro prox*)

Abaixo nós temos a segunda versão do nosso algoritmo

```
Se (p->prox != Nulo)                                // existe a posição 1?
    q = p->prox                                       // a 1o posição é a posição 0
    Enquanto ( q != Nulo )
        Imprime(q->num)
        q = q->prox                                  //
    Se (q != Nulo)                                   // pulando de 2 em 2
        q = q->prox                                  //
```