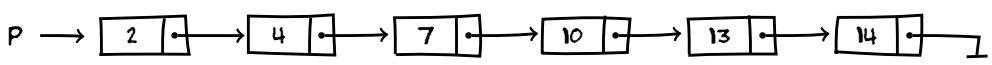
A operação de atualização

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



A tarefa é

→ *encontrar o elemento* `x`

modificar o seu valor para `y` *e recoloca-lo na posição correta*

Bom, aqui a própria tarefa já foi especificada em etapas.

A primeira e a segunda etapas são fáceis.

E a terceira etapa pode ter dois casos

- se $y > x$, o elemento pode ir para frente
- se $y < x$, o elemento pode ir para trás

O primeiro caso é fácil, porque ele consiste em fazer uma inserção na lista ordenada — (*nós podemos chamar uma função para fazer isso*)

E no segundo caso, a gente carrega o elemento para trás — (*utilizando o valor de retorno da função*)

O lugar em que a inserção vai ser feita é o primeiro elemento menor que `y` — (*se houver algum*)

Ou então a primeira posição da lista — (*se não houver nenhum*)

A coisa fica assim

```
func atualização-Rec (q, x, y, pri)

      Se ( q == Nulo  OU  q->num > x )                  // o x não está lá ...
            Retorna (Nulo, Nulo)

      Sse (q->num == x )
            q->num = y;   q->prox = Nulo
            Se (y > x)
                  px = inserçãoOrdP-Rec (q->prox, q)
                  Retorna (px, Nulo)
            Senão
                  Retorna (q->prox, q)

      Senão
            px,el = atualização-Rec (q->prox, x, y, 0)
            Se (el != Nulo  E  q->num < el->num)
                  el->prox = q->prox
                  q->prox = el
                  Retorna (q, Nulo)
            Senão
                  el->prox = q
                  Retorna (el, Nulo)
```

A função `main()` fica assim

```
func main ()

      p,el = atualização-Rec (p, 10, 17 1)
```

— (*e a função* `inserçãoOrdP-Rec()` *fica como exercício*)

◇