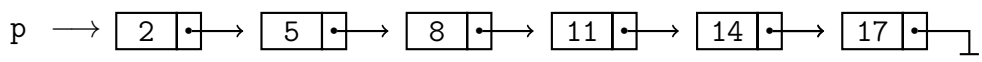
Imagine que o ponteiro `p` aponta para uma lista encadeada ordenada



A tarefa é

→ *encontrar o elemento* `x`
modificar o seu valor para `y`
e colocá-lo no seu lugar correto na lista ordenada

Aqui mais uma vez, a gente precisa fazer a coisa em etapas

1. encontrar o elemento `x`
2. atualizar o seu valor e remove-lo da lista
3. colocar o registro no lugar correto

Para encontrar e remover o elemento, a gente vai utilizar a técnica de percorrer a lista com dois ponteiros.

Daí, se `y` > `x`, a gente continua percorrendo a lista para encontrar o novo lugar do registro.

E se `y` < `x`, a gente precisa percorrer a lista desde o início outra vez.

A coisa fica assim

```
func opAtualização ( x, y, p )

      q = p;   qant = Nulo;   achei = 0                    //
      Enquanto (q != Nulo )                                //
            Se ( q->num = x )                              // 1a Etapa
                  achei = 1;   Quebra                      //
            qant = q;   q = q->prox                        //
      Se ( achei == 0 )   Retorna (p)

      Se ( qant == Nulo )   p = p->prox                    //
      Senão                 qant-prox = q->prox            // 2a Etapa
      q->num = y                                           //

      Se ( y > x )                                         //
            Enquanto ( qant->prox != Nulo)                 //
                  Se ( (qant-prox)->num > y )              // 3a Etapa (a)
                        Quebra                             //
                  qant = qant->prox                        //
            q->prox = qant->prox                           //
            qant->prox = q                                 //

      Senão                                                //
            Se ( p->num > y )                              //
                  q->prox = p                              // 3a etapa (b)
                  p = q                                    //


       Senão
             qant = p
             Enquanto ( qant->prox != Nulo)
                   Se ( (qant-prox)->num > y )
                         Quebra
                   qant = qant->prox
             q->prox = quant->prox
             qant->prox = q

Retorna (p)
```

◇