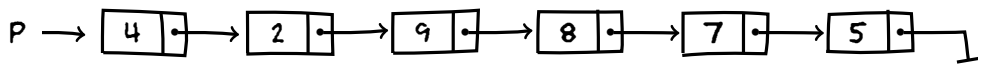
O maior de todos

Imagine que o ponteiro p aponta para uma lista encadeada



A tarefa é

→ *encontrar o maior elemento da lista*

Bom, aqui a gente pode fazer a coisa na ida ou na volta.

Fazendo as coisas na ida, a gente utiliza um parâmetro `maa` que lembra qual foi o maior valor que foi visto até aqui.

No início, a função `main()` atribui o valor -1 a esse parâmetro, para indicar que ainda não foi visto nenhum elemento.

A coisa fica assim

```
func maior-Rec (q,maa)
      Se ( q == Nulo )   Retorna                    // já acabei ...
      Se ( q->num > maa )                           // eu sou o maior até aqui?
            resp = maior-Rec (q->prox,q->num)
      Senão
            resp = maior-Rec (q->prox,maa)
      Retorna (resp)
```

E a função `main()` fica assim

```
func main ()
      M = maior-Rec (p,-1)
      Imprime ('o maior elemento é', M)
```

Legal.

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

Quer dizer, a gente também pode fazer as coisas na volta.

E a gente pode retornar um ponteiro para o maior elemento.

Fazendo as coisas assim, a gente primeiro vai até o fim da lista.

Daí, a chamada que vê o `Nulo` retorna o ponteiro `Nulo`.

Todas as outras chamadas comparam o seu elemento com aquele que está sendo retornado, e retornam o maior dos dois.

A coisa fica assim

```
func maior2-Rec (q,maa)
      Se ( q == Nulo )   Retorna                    // já acabei ...
      resp = maior2-Rec (q->prox)
      Se (resp == Nulo  OU  q->num > resp->num)     // o último ou o maior até aqui
            Retorna (q)
      Senão
            Retorna (resp)
```

E a função `main()` fica assim

```
func main ()
      pM = maior2-Rec (p)
      Se (pM != Nulo)
            Imprime ('o maior elemento é', pM->num)
```

◇