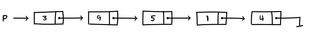
Inserção no fim da lista

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

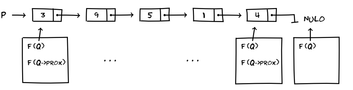


A tarefa é

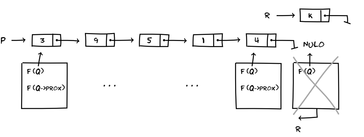
→ *inserir o número* `k` *no fim da fila*

Bom, nós queremos realizar essa tarefa com um algoritmo recursivo.

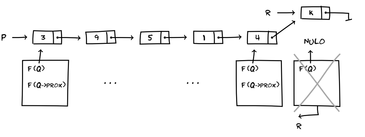
Então, a ideia é que as chamadas recursivas vão alcançar o fim da lista



Daí, a chamada que vê o `Nulo` cria um novo registro, coloca o número `k` dentro dele, e retorna o ponteiro para o registro para chamada anterior



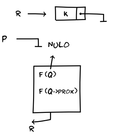
E é a chamada anterior que tem a responsabilidade de conectar o novo registro ao último elemento da lista



Daí, como a operação de inserção já foi realizada, todas as demais chamadas retornam o ponteiro `Nulo` — (*indicando para a chamada anterior que ela não tem nada a fazer*)

Mas, é preciso ter cuidado com o caso da lista vazia

Nesse caso, a primeira chamada recursiva já cria o novo registro e retorna um ponteiro para ele



Daí, quando a função `main()` recebe um ponteiro diferente de `Nulo`, ela atualiza o ponteiro `p`

A coisa fica assim

```
func inserçãoFim-Rec (q,k)

    Se ( q == Nulo )
          r = Novo (RegInt)                        //
          r->num,prox = k,Nulo                     // criação do registro
          Retorna (r)                              //
    Senão
          resp = inserçãoFim-Rec (q->prox,k)
          Se (resp != Nulo)
               q->prox = resp                      // conexão com o último
          Retorna (Nulo)
```

E a função `main()` fica assim

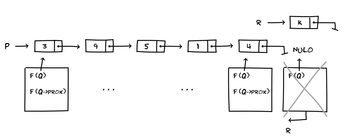
```
func main ()

     resp = inserçãoFim-Rec (p,k)
     Se ( resp != Nulo )
          p = resp
```

Legal

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

Quer dizer, a gente pode simplificar um pouco as coisas utilizando o truque da *reconexão*.

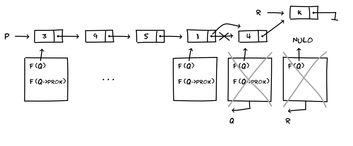
Vamos olhar outra vez para essa situação



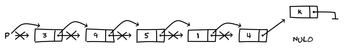
Aqui, a chamada que vê o último elemento está recebendo um ponteiro para o novo registro, para conectá-lo à lista.

Só que dessa vez, ao invés de retornar `Nulo` para a chamada anterior ela vai retornar o seu próprio ponteiro `q` — (*i.e., um ponteiro para o elemento 4*)

Daí, a chamada que vê o penúltimo elemento vai imaginar que o 4 acabou de ser inserido na lista, e vai se (re)conectar a ele



E agora, a coisa continua assim até o fim, com cada elemento sendo reconectado ao próximo



A coisa fica assim

```
func inserçãoFim2-Rec (q,k)

     Se ( q == Nulo )
          r = Novo (RegInt)                        //
          r->num,prox = k,Nulo                     // criação do registro
          Retorna (r)                              //
     Senão
          resp = inserçãoFim2-Rec (q->prox,k)
          q->prox = resp                           // re-conexão
          Retorna (q)
```

E a função `main()` fica assim

```
func main ()

     p = inserçãoFim2-Rec (p,k)
```

Legal, não é?