A pilha

A esperteza que nós acabamos de ver é uma estratégia de computação que é usada em um monte de situações diferentes.

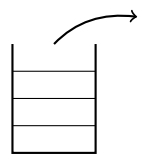
Quer dizer, essa é a situação em que nós temos várias coisas para fazer, mas nem todas podem ser feitas imediatamente — (*algumas precisam aguardar que outras sejam feitas primeiro*)

E daí, a esperteza consiste em utilizar uma pilha de coisas que ainda precisam ser feitas — (*ou uma pilha de tarefas*)

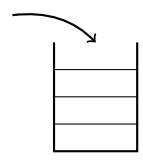
Foi por isso que nós demos o nome de P para a nossa lista auxiliar.

Agora, duas características importantes de uma pilha de tarefas são

1. a gente sempre pega a tarefa no topo da pilha para realizar em seguida



2. a gente sempre coloca a uma nova tarefa no topo da pilha



Essas operações são realizadas tão frequentemente, que é útil ter funções que as implementam

```
func Push ( elem, P )
     topo++
     P[topo] = elem
```

```
func Pop ( P )
     elem = P[topo]
     topo--
     Retorna ( elem )
```

Além disso, também é útil ter uma função que apenas inspeciona o elemento do topo, sem removê-lo

```
func Peep ( P )
     elem = P[topo]
     Retorna ( elem )
```

E é útil ter uma função que nos diz qual é o tamanho da pilha

```
func Size ( P )
     Retorna ( topo )
```

Certo.

Agora que nós temos essa nova estrutura de dados, é uma boa ideia reescrever o código do nosso avaliador de expressões

```
func CECSP ( V, i, P )

    i++                                          // pula o abre-parênteses
    zero = Size(P)                               // imagina que P está vazia

    // 1a ETAPA: calcular as multiplicações

    Enquanto ( V[i] != fecha-parênteses )

    Se ( V[i] == abre-parênteses )    CECSP ( V, i, P )

    Senão                             Push ( V[i], P )

    Se ( V[i] == multiplicação )

       i++

       Se ( V[i] == abre-parênteses )    CECSP ( V, i, P )

       Senão                             Push ( V[i], P )
                                         i++

       N2 = Pop ( P )
       op = Pop ( P )
       N1 = Pop ( P )

       res = Calcula ( N1, op, N2 )

       Push ( res, P )

    Senão

       i++


    // 2a ETAPA: calcular as somas

    Enquanto ( j > zero + 1 )

        N2 = Pop ( P )
        op = Pop ( P )
        N1 = Pop ( P )

        res = Calcula ( N1, op, N2 )

        Push ( res, P )

    i++                                          // pula o fecha-parênteses
```