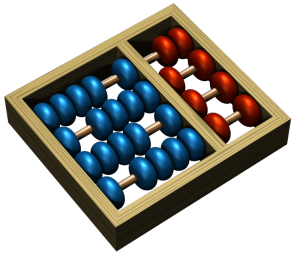
MC202 - Estruturas de Dados

Emilio Francesquini
francesquini@ic.unicamp.br

Instituto de Computação - UNICAMP

Aula 04 - 14 de março de 2017



UNICAMP

Disclaimer

- Esses slides foram preparados para um curso de Estrutura de Dados ministrado na UNICAMP
- Este material pode ser usado livremente desde que sejam mantidos os créditos dos autores e da instituição.
- Muitos dos exemplos apresentados foram retirados de materiais preparados pelos Profs. T. Kowaltowski e O. Lee da UNICAMP assim como do Prof. P. Feofiloff do IME-USP

Pilhas

Pilhas - Intuição

- Todos tem uma noção intuitiva do que é uma pilha (=*stack*)
- É prático
  - Verificar qual é o item no topo da pilha
  - Colocar algo (=*empilhar*) no topo (=*top*) da pilha
  - Tirar algo (=*desempilhar*) do topo da pilha
    - Um ou mais elementos
- É mais trabalhoso
  - Retirar um (ou alguns) itens intermediários
  - Verificar o conteúdo de um elemento que não está no topo

UNICAMP

Pilhas

- Pilhas são uma manifestação do que chamamos de tipo abstrato de dados (=*Abstract Data Type*)
- Nem falamos sobre a implementação, mas já sabemos que para ser util ela terá as seguintes operações
  - vazio (=*empty*) – Verifica se a pilha está vazia
  - empilhar (=*push*) – Empilha um elemento
  - desempilhar (=*pop*) – Desempilha um elemento
  - consultar (=*peek*) – Olha o primeiro elemento da lista sem retirá-lo
- Tradicionalmente os nomes em inglês das operações costumam ser mais utilizados
  - Mas por alguma razão o nome stack não é tão utilizado assim...

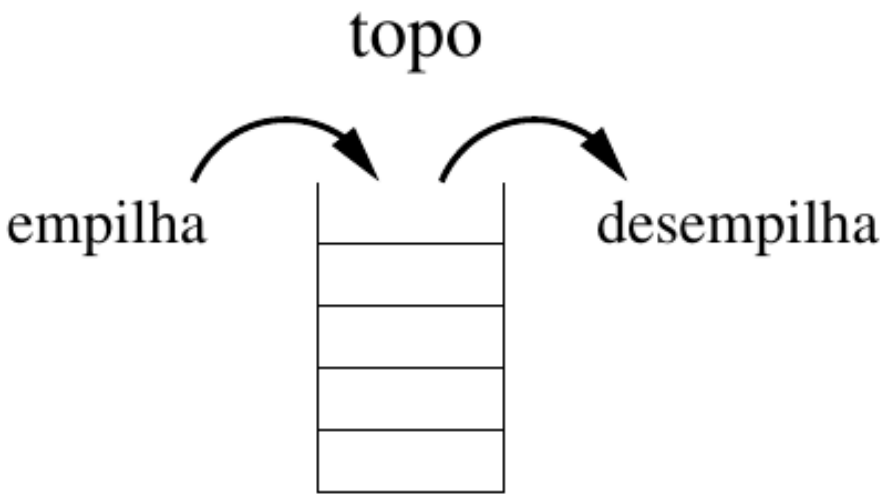
UNICAMP

Pilhas - Operação

- As seguintes frases resumem o comportamento de uma pilha:
  - *o elemento removido da pilha é o que está lá há menos tempo*
  - *o primeiro elemento inserido na pilha é o último a ser removido.*

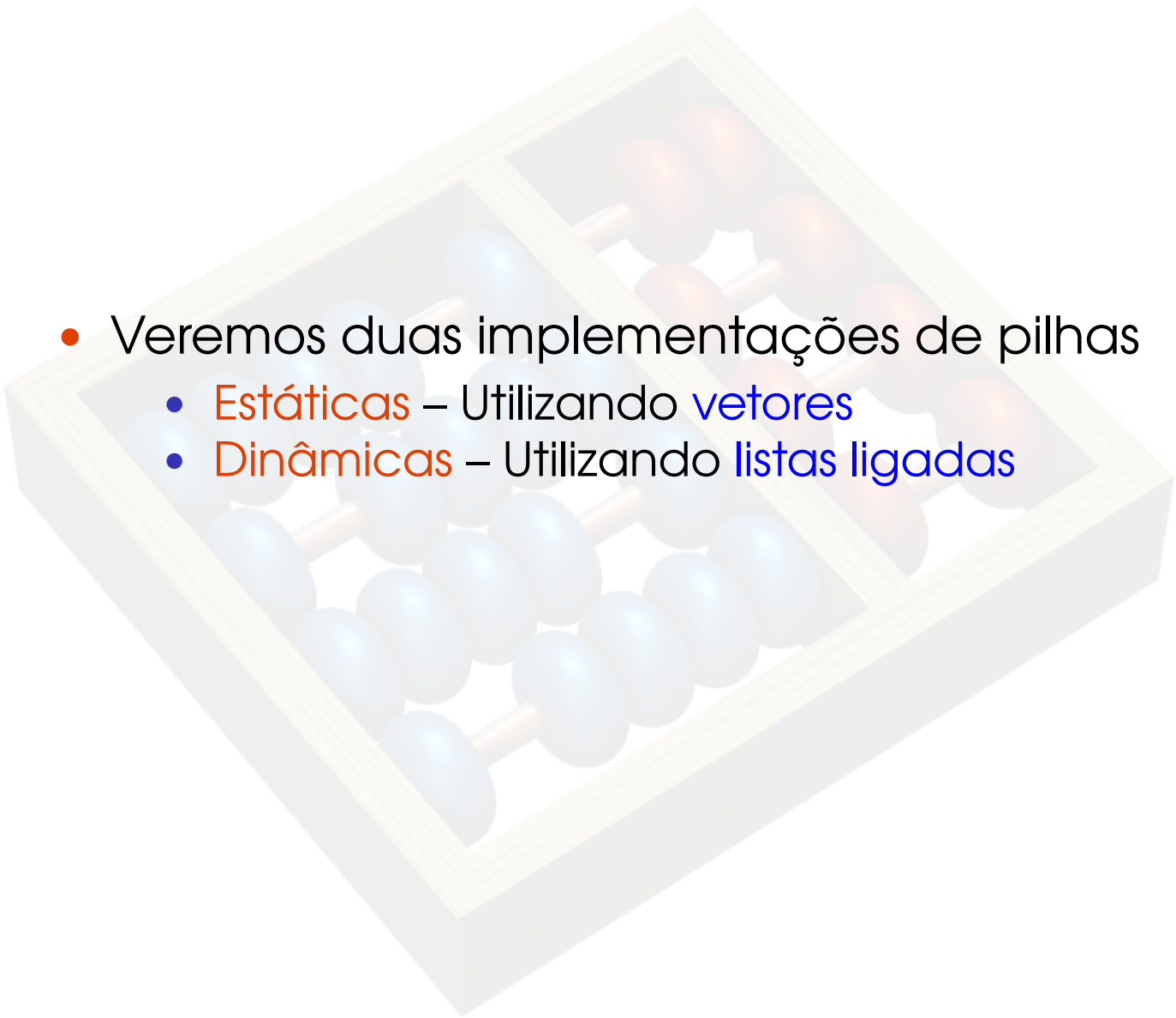
Esta politica de operação é chamada de LIFO (Last-In-First-Out)
Quem entrou antes, saí depois...

Pilhas - Resumindo



- Topo (top): é o lado em que as operações são feitas;
- empilhar (push) = inserir no topo;
- desempilhar (pop) = remover do topo.

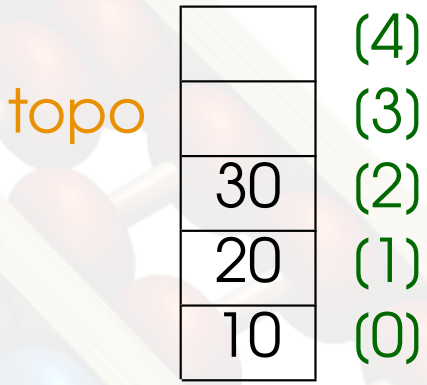
Pilhas - Implementação



- Veremos duas implementações de pilhas
  - Estáticas – Utilizando vetores
  - Dinâmicas – Utilizando listas ligadas

UNICAMP

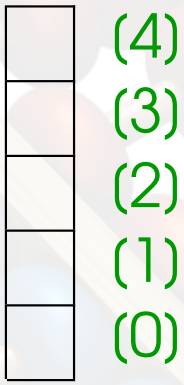
Pilhas Estáticas



- Neste caso o tamanho máximo da pilha é n = 5
- topo sempre aponta para o próximo elemento vazio
  - A pilha está cheia se o topo = n
  - A pilha está vazia se o topo = 0

UNICAMP

Pilhas Estáticas - Exemplo



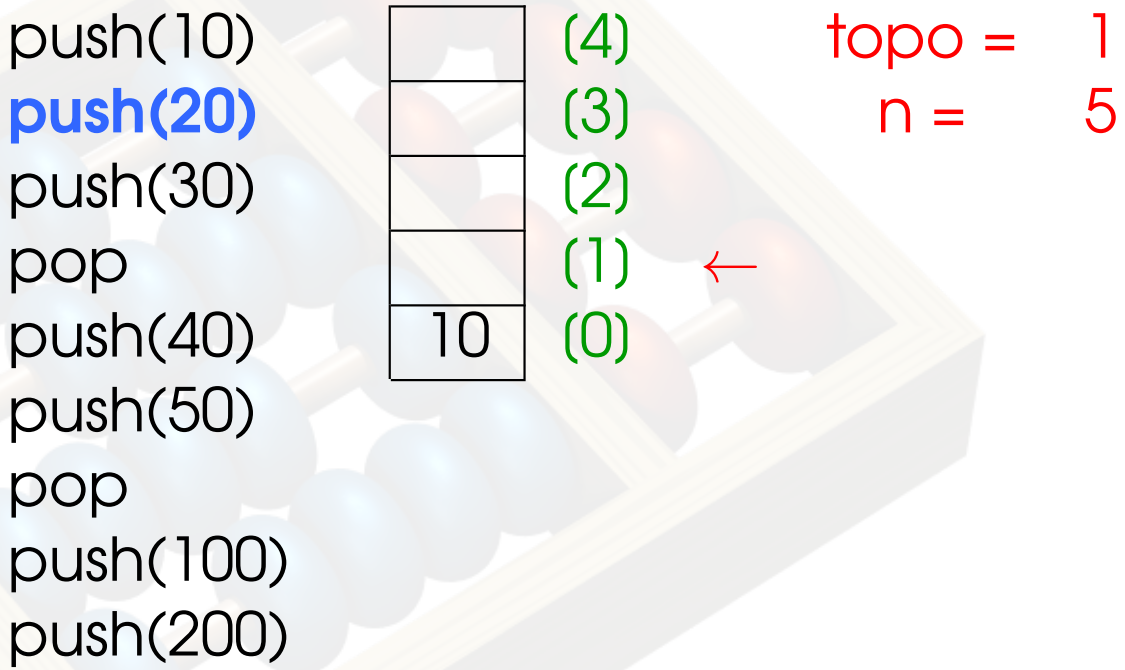
push(10) (4)
push(20) (3)
push(30) (2)
pop (1)
push(40) (0) ←
push(50)
pop
push(100)
push(200)

n = 5
topo = 0

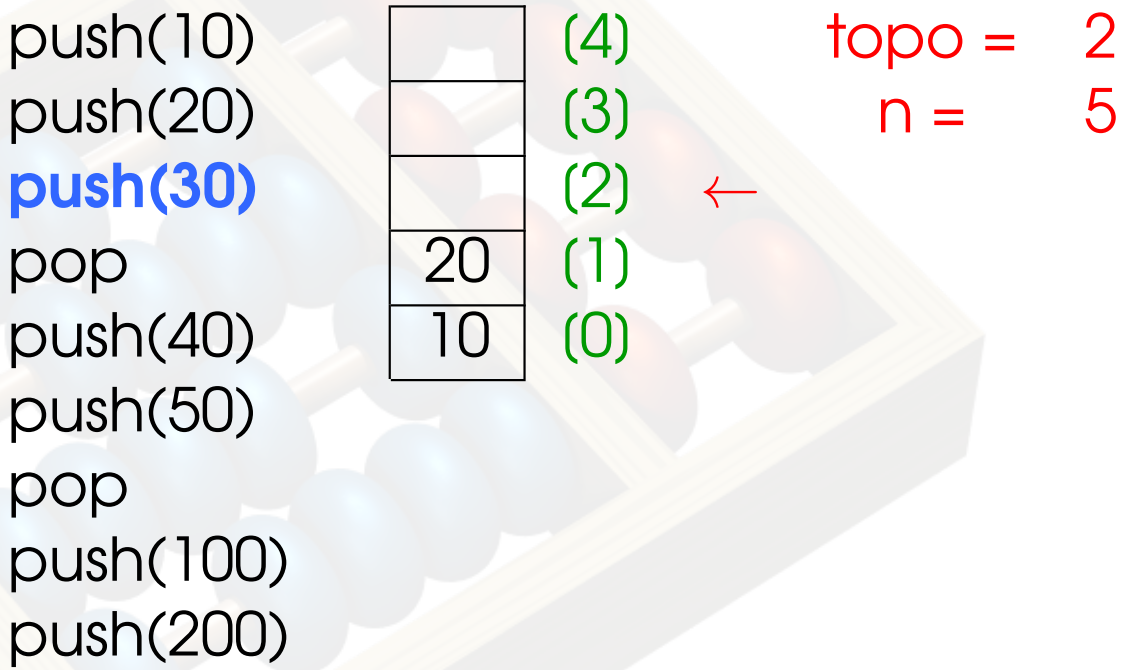
topo == 0
Pilha vazia!

UNICAMP

Pilhas Estáticas - Exemplo



Pilhas Estáticas - Exemplo



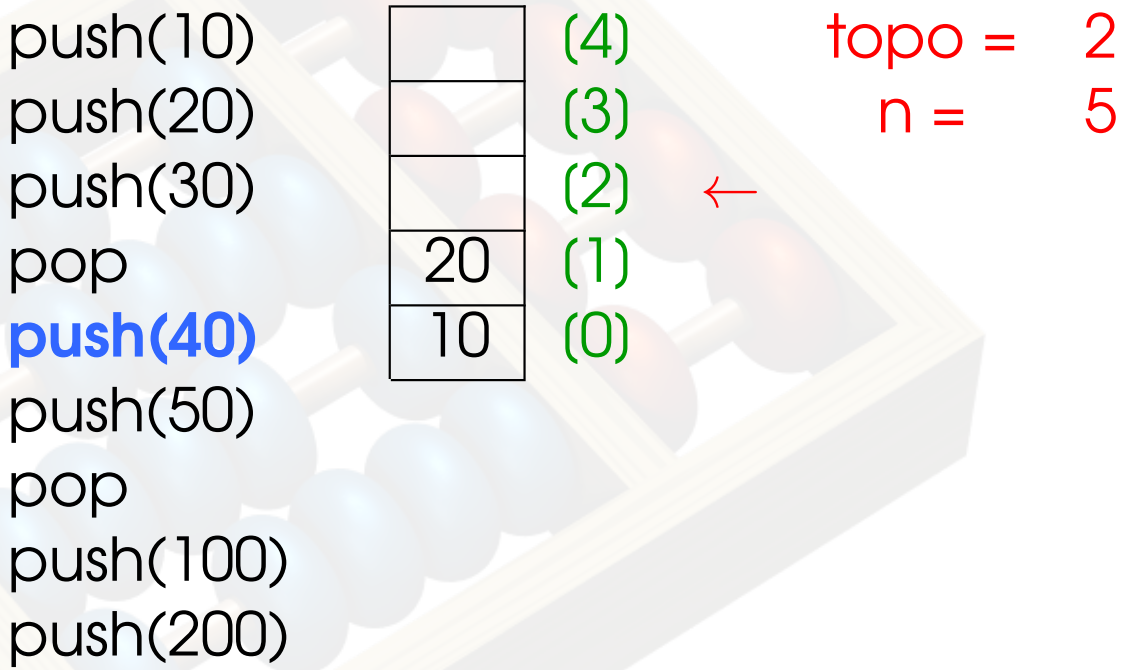
Pilhas Estáticas - Exemplo

push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)

topo = 3
n = 5

UNICAMP

Pilhas Estáticas - Exemplo

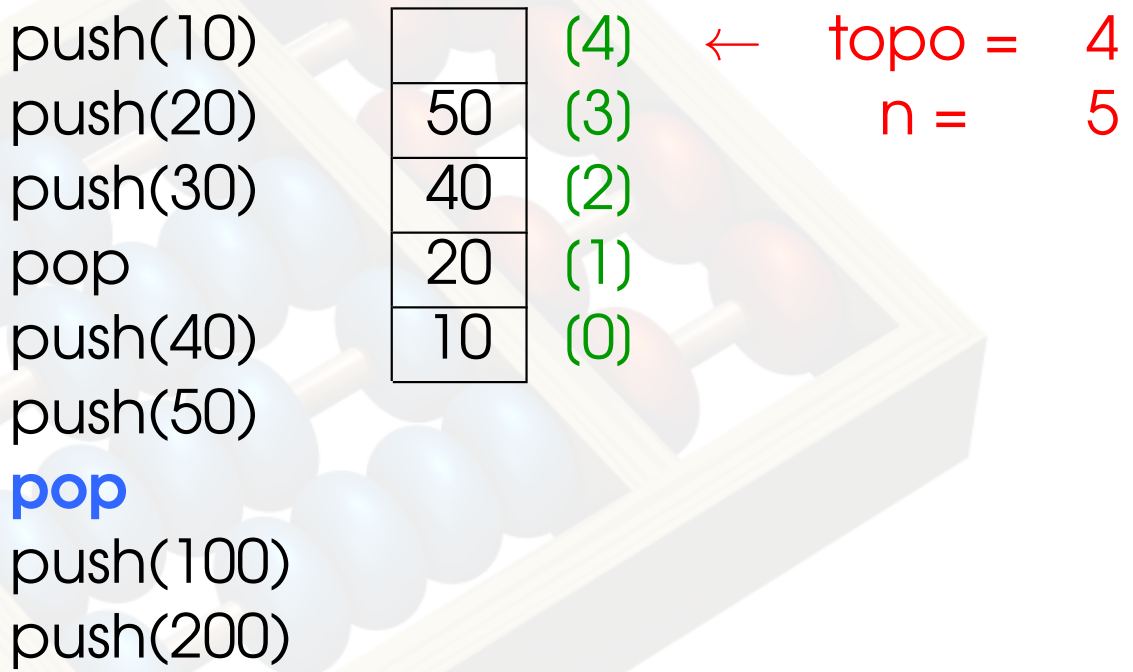


Pilhas Estáticas - Exemplo

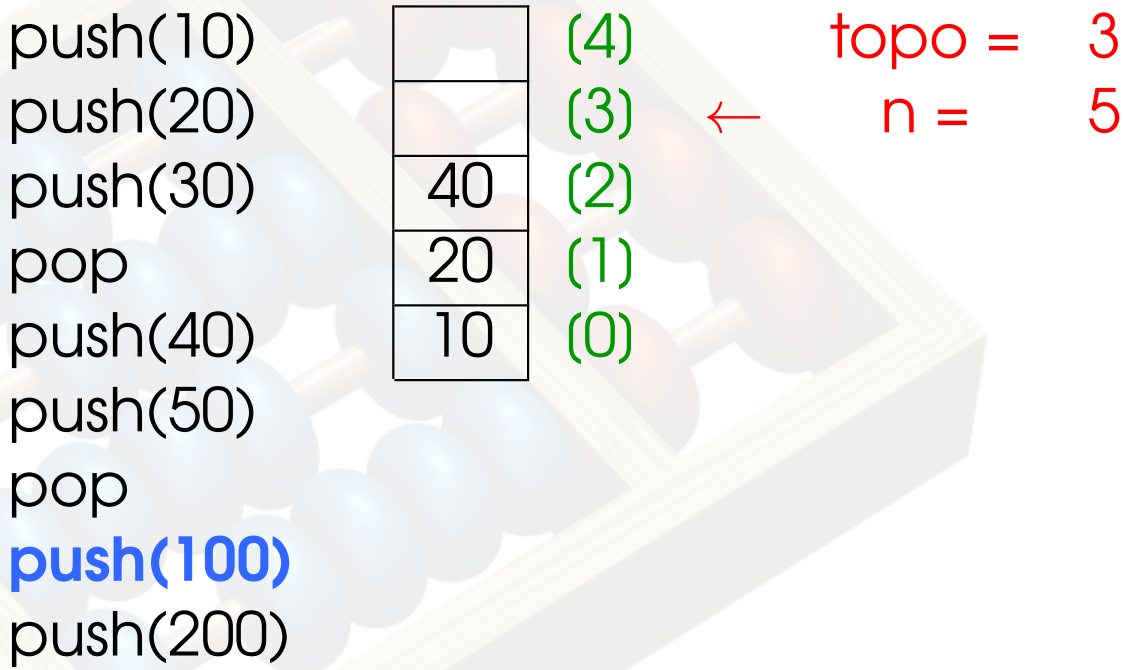
push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)

topo = 3
n = 5

Pilhas Estáticas - Exemplo

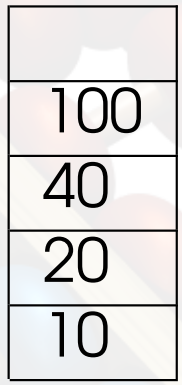


Pilhas Estáticas - Exemplo



Pilhas Estáticas - Exemplo

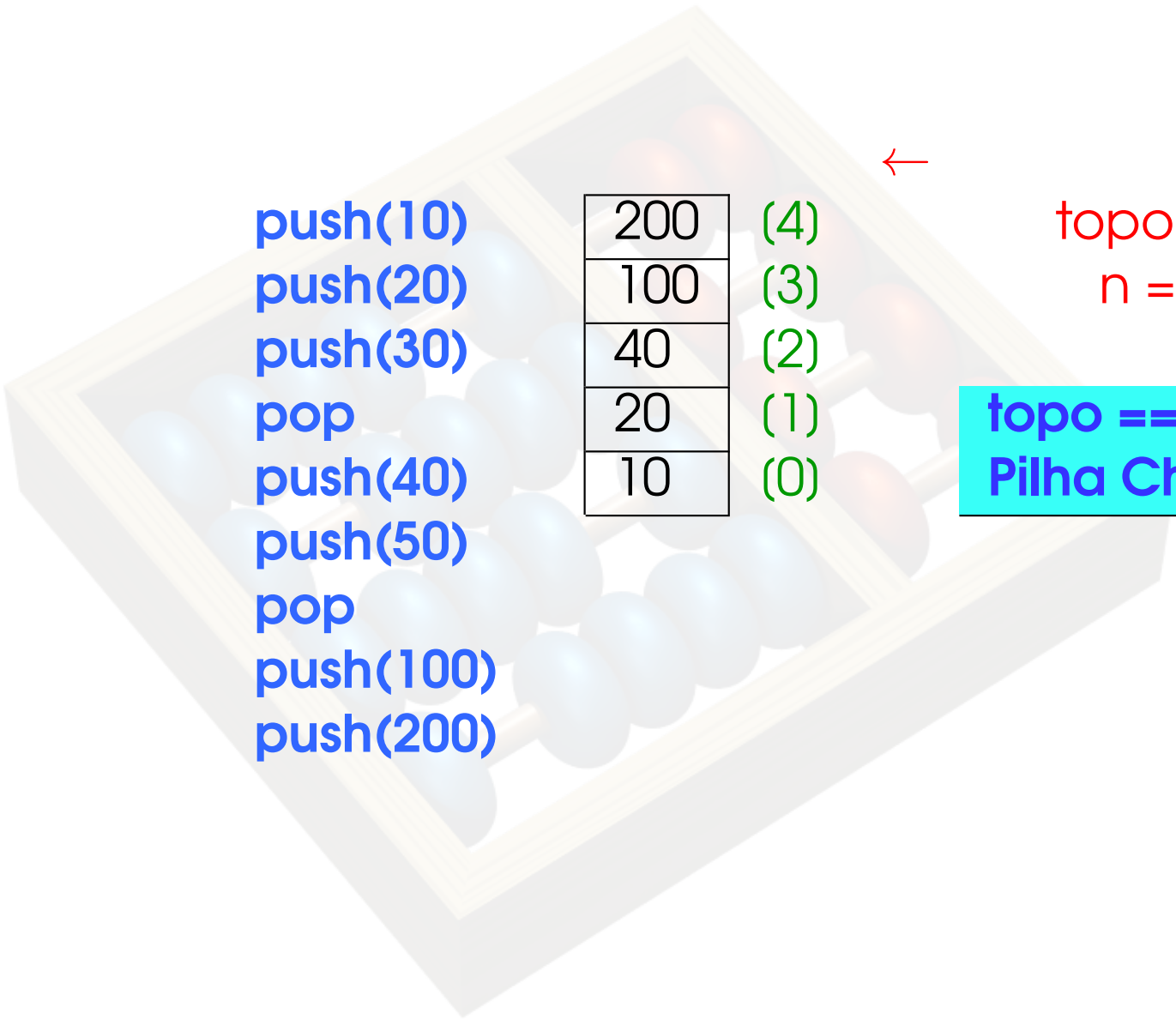
push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)



← topo = 4
n = 5

UNICAMP

Pilhas Estáticas - Exemplo



Manipulando Pilhas Estáticas

- Empilhar (push) – `pilha[topo++] = x;`
- Consultar (peek) – `x = pilha[topo - 1];`
- Desempilhar (pop) – `x = pilha[--topo];`
- Vazio (empty) – `topo == 0`
- Cheio (full) – `topo == n`

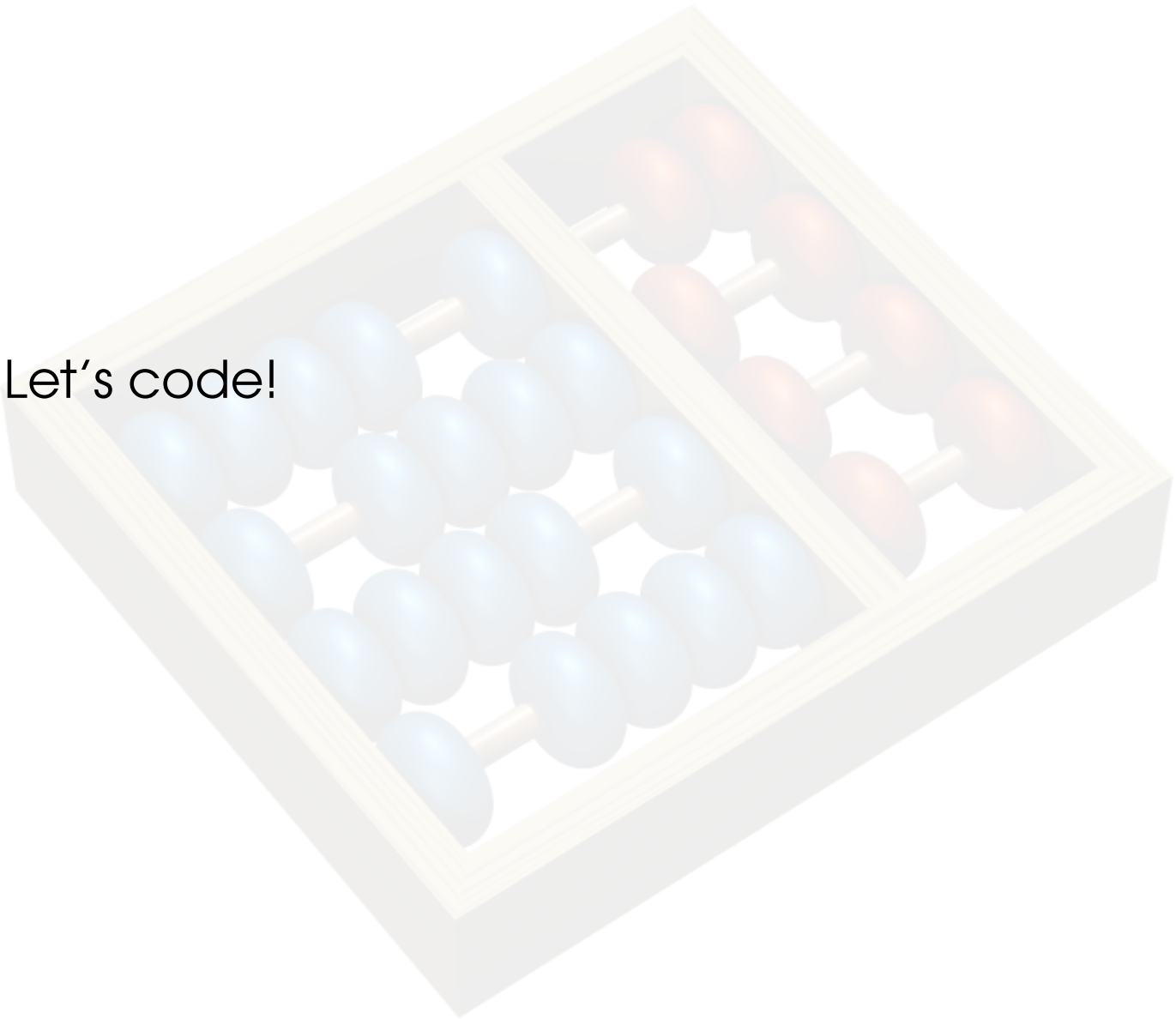
E se em algum momento da execução `topo > n`?

Manipulando Pilhas Estáticas

- Empilhar (push) – `pilha[topo++] = x;`
- Consultar (peek) – `x = pilha[topo - 1];`
- Desempilhar (pop) – `x = pilha[--topo];`
- Vazio (empty) – `topo == 0`
- Cheio (full) – `topo == n`

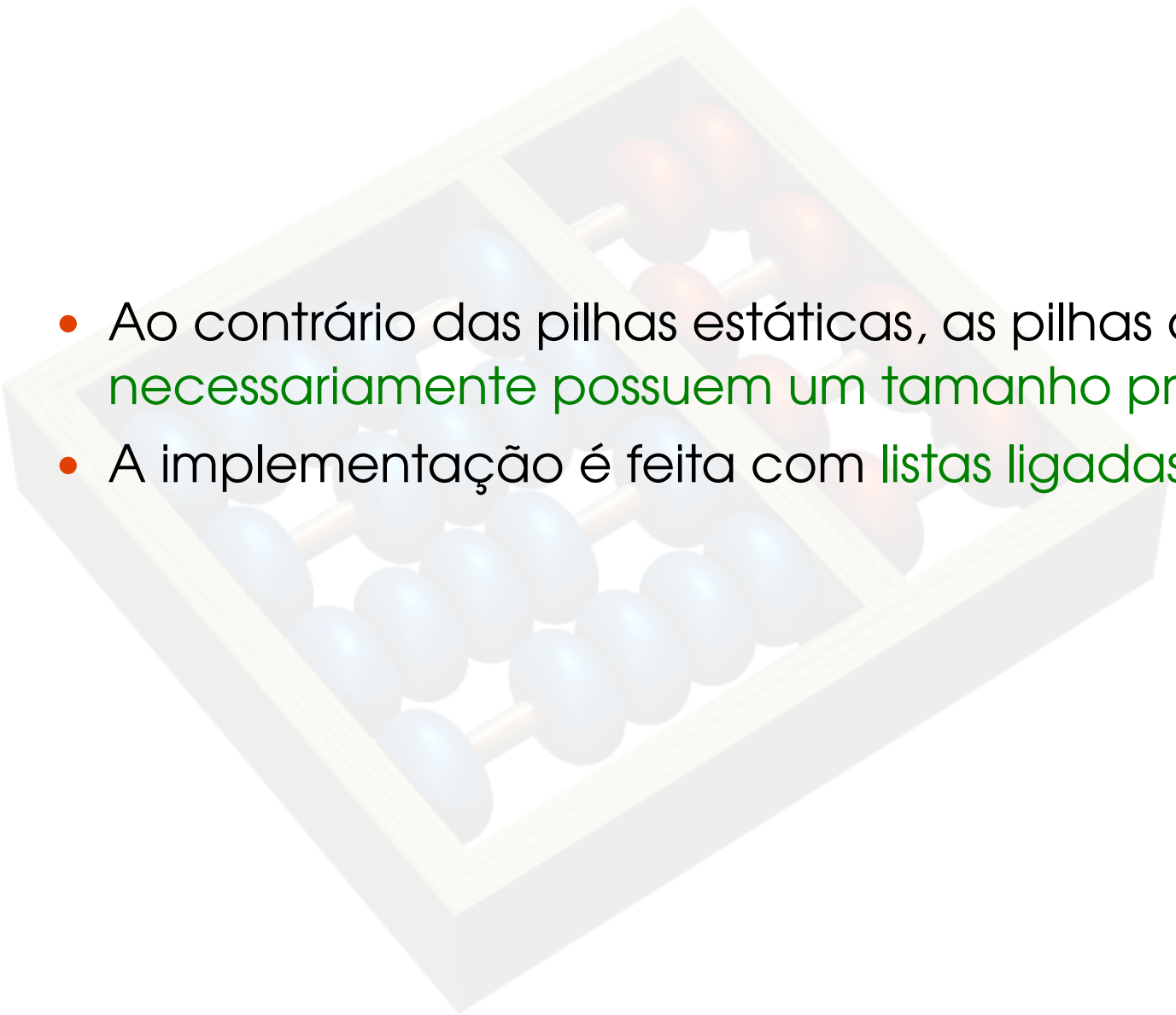
E se em algum momento da execução `topo > n`?
BUG!

Mão na massa!



Let's code!

Pilhas dinâmicas



- Ao contrário das pilhas estáticas, as pilhas dinâmicas não necessariamente possuem um tamanho pré-definido
- A implementação é feita com listas ligadas

Pilhas dinâmicas - Exemplo

push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)

Pilhas dinâmicas - Exemplo

push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)

UNICAMP

Pilhas dinâmicas - Exemplo

push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)

UNICAMP

Pilhas dinâmicas - Exemplo

push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)

UNICAMP

Pilhas dinâmicas - Exemplo

push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)

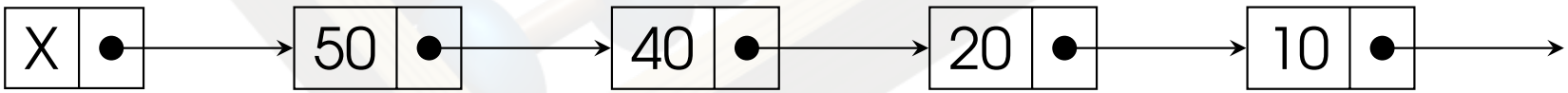
UNICAMP

Pilhas dinâmicas - Exemplo

push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)

Pilhas dinâmicas - Exemplo

push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)



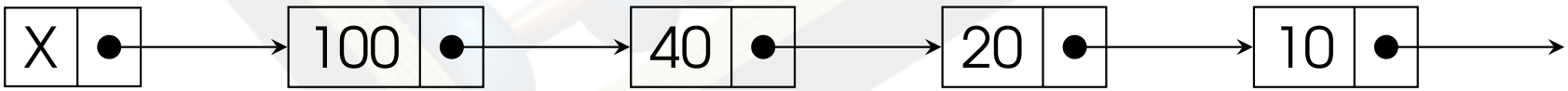
UNICAMP

Pilhas dinâmicas - Exemplo

push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)

Pilhas dinâmicas - Exemplo

push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)



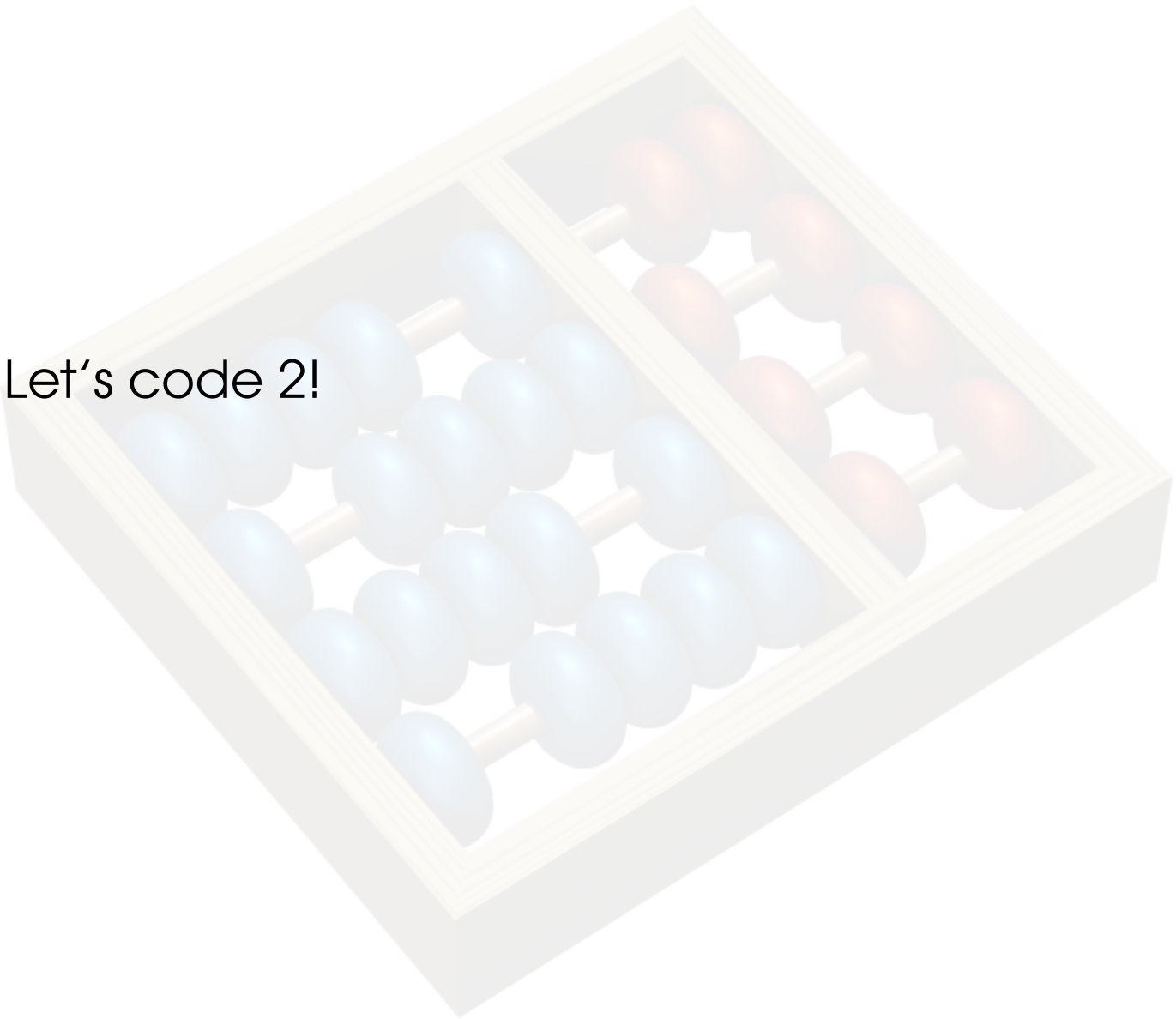
UNICAMP

Pilhas dinâmicas - Exemplo

push(10)
push(20)
push(30)
pop
push(40)
push(50)
pop
push(100)
push(200)

UNICAMP

Mão na massa!



Let's code 2!

UNICAMP

Balanceamento de parênteses

Problema clássico: verificar se uma sequência de (,),[e] é balanceada ou bem-formada.

```
()
([])
([)]
([()]())
)(
([
([[(][()])])
[([])[([])]
(([((()([](([]))[]))])))
```

Como resolver este problema?

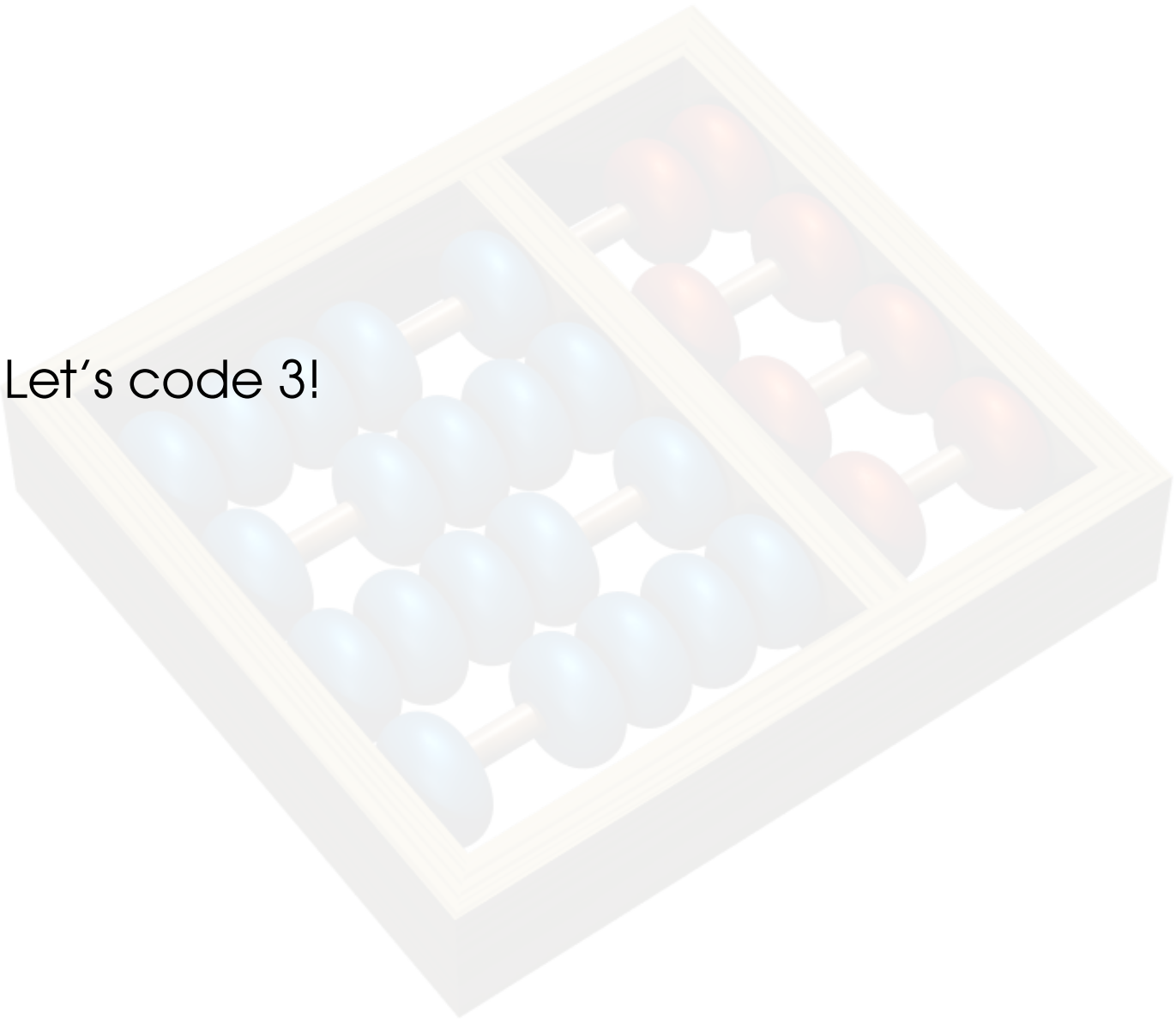
Balanceamento de parênteses

- Comece com a pilha vazia;
- Enquanto tiver símbolos na entrada
  - Leia o próximo símbolo x da entrada;
  - Se x = (ou x = (então empilhe x;
  - Se x =) ou x =) então
    veja se o que está no topo casa com x;
  - Se SIM então desempilhe;
  - Senão Não é balanceada;
- Verifique se a pilha está vazia;
- Se SIM então Está balanceada;
- Senão Não é balanceada;

Balanceamento - Exemplo

Entrada	Pilha
([([ ] [( )])])	
[([ ] [( )])])	(
([ ] [( )])])	([
[ ] [( )])])	([(
] [( )])])	([([
[( )])])	([(
( )])])	([([
)])])	([([(
])])	([([
)])	([(
])	([
)	(

Mão na massa!



Let's code 3!

UNICAMP