



UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS - CSHNB
CURSO DE SISTEMAS DE INFORMAÇÃO
PICOS - PI

PILHAS (ESTÁTICAS E DINÂMICAS)

Prof. Ma. Luana Batista da Cruz
luana.b.cruz@nca.ufma.br

Roteiro

2

- Introdução
- Pilhas de dados
- Pilha estática
- Pilha dinâmica

3

Introdução

Definição de pilha sequencial

Introdução

4

- ❑ **O que são pilhas sequenciais?**
 - São estruturas de dados que armazenam os elementos em um formato sequencial, empilhando um item acima do outro

Introdução

5

❑ O que são pilhas sequenciais?

- São estruturas de dados que armazenam os elementos em um formato sequencial, empilhando um item acima do outro



Introdução

6

❑ O que são pilhas sequenciais?

- São estruturas de dados que armazenam os elementos em um formato sequencial, empilhando um item acima do outro



Introdução

7

- **Ordem de retirada x Ordem de armazenamento**
 - É comum: ordem de remoção → armazenado
 - Exemplo: organizando entregas de pizza



Introdução

8

- O que ocorre na pizzeria?



Introdução

9

- O que ocorre na pizzeria?



Introdução

10

- O que ocorre na pizzeria?



Introdução

11

- O que ocorre na pizzeria?



Introdução

12

- O que ocorre na pizzeria?



Introdução

13

- O que ocorre na pizzeria?



Introdução

14

- O que ocorre na pizzeria?

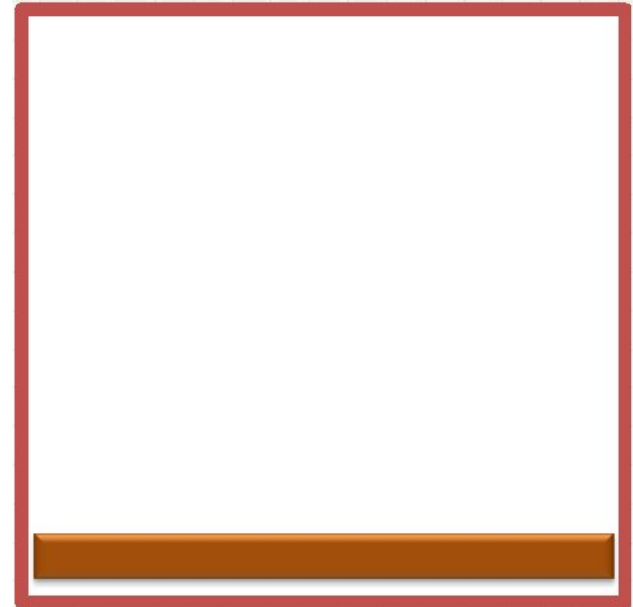


...

Introdução

15

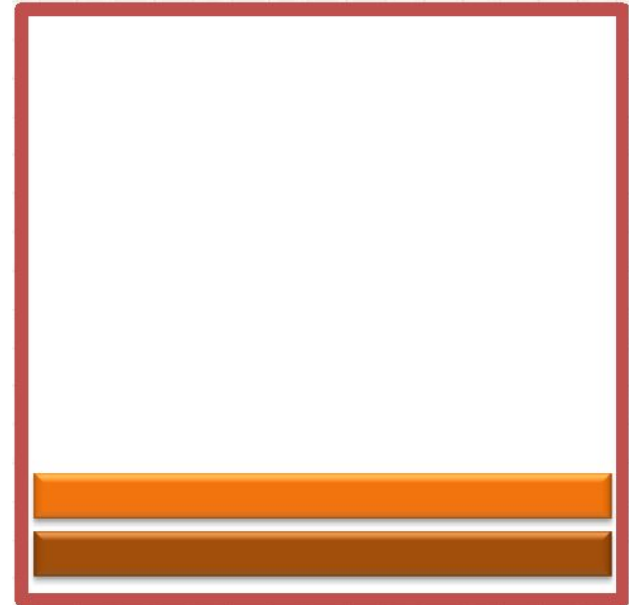
- Como os pedidos vão sendo **armazenados** para entrega?



Introdução

16

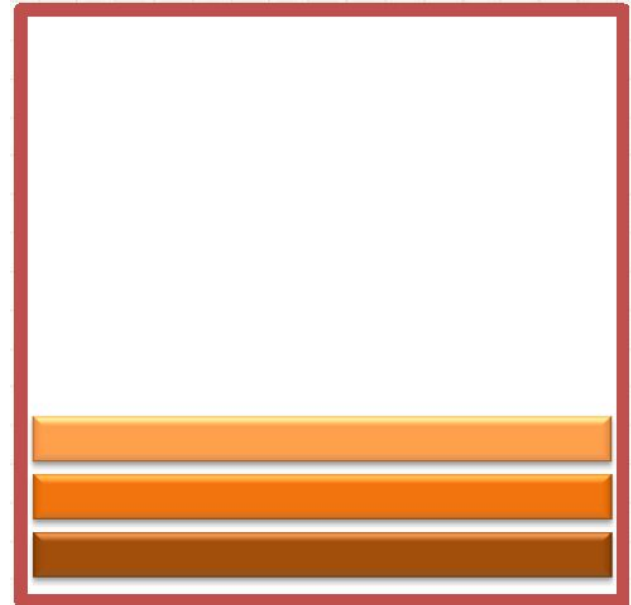
- Como os pedidos vão sendo **armazenados** para entrega?



Introdução

17

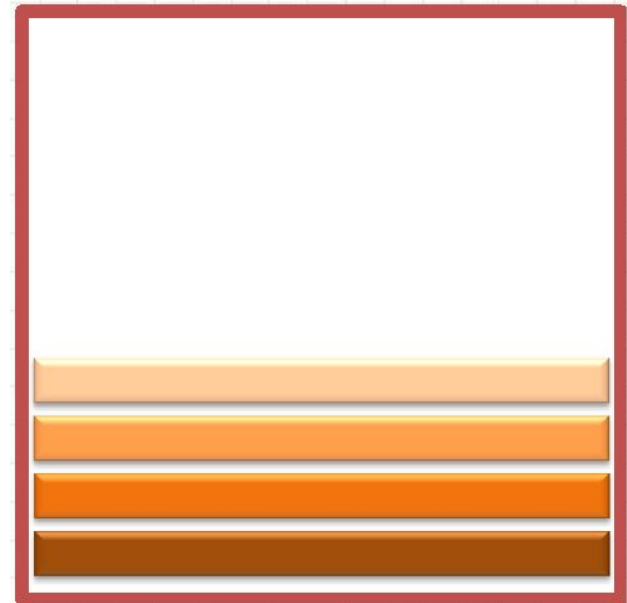
- Como os pedidos vão sendo **armazenados** para entrega?



Introdução

18

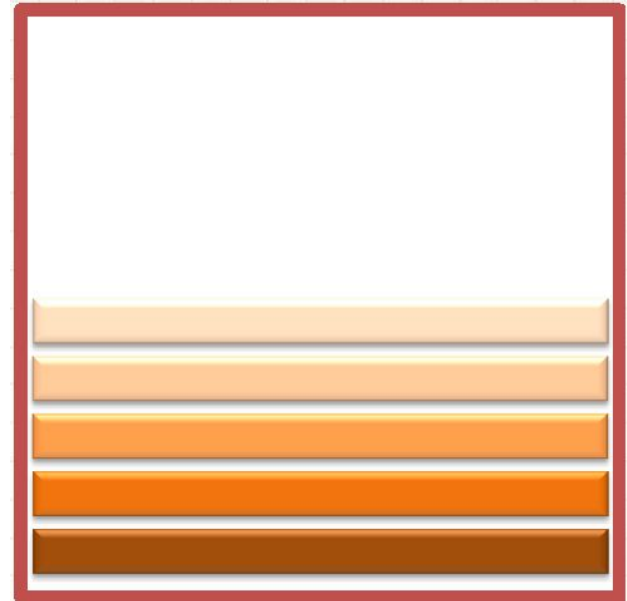
- Como os pedidos vão sendo **armazenados** para entrega?



Introdução

19

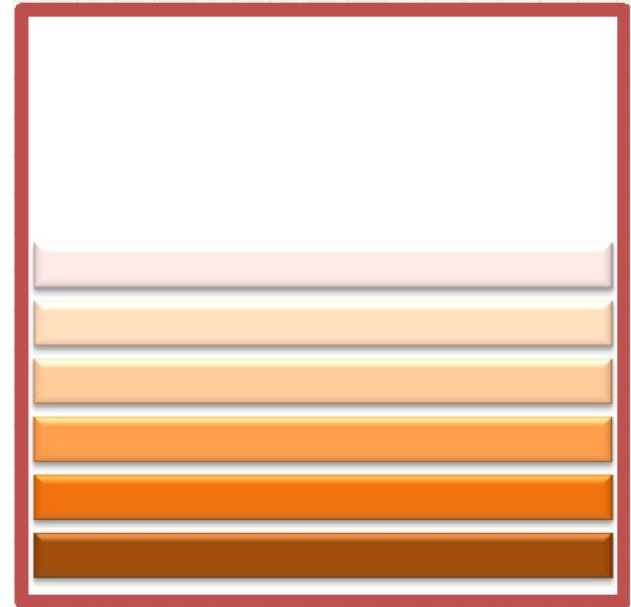
- Como os pedidos vão sendo **armazenados** para entrega?



Introdução

20

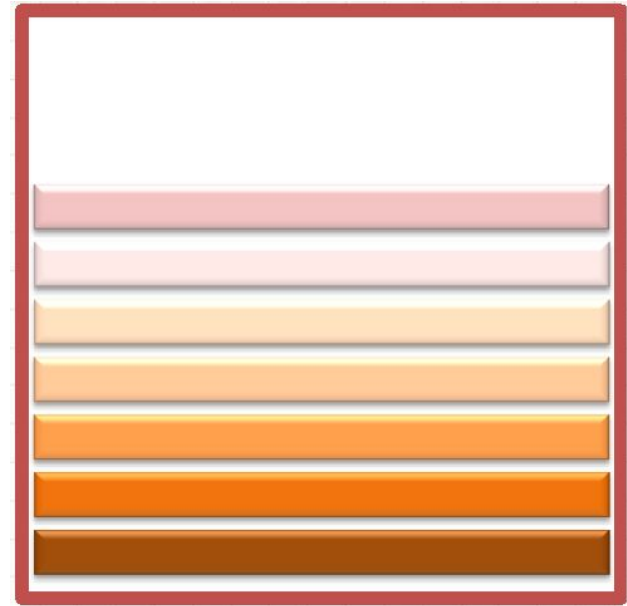
- Como os pedidos vão sendo **armazenados** para entrega?



Introdução

21

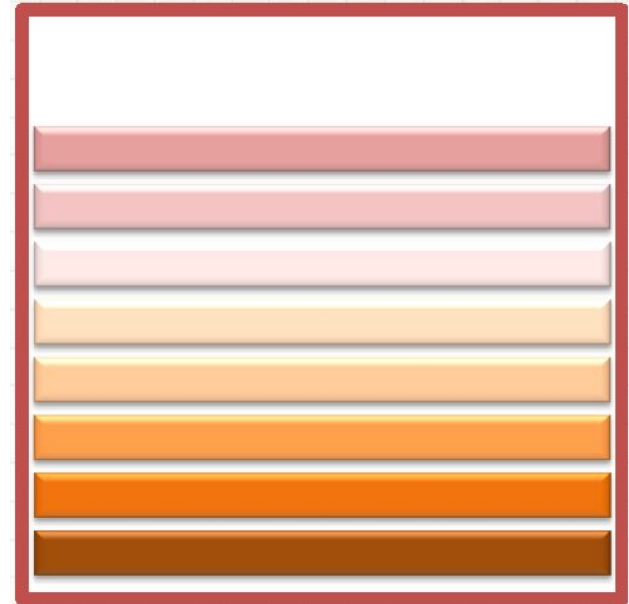
- Como os pedidos vão sendo **armazenados** para entrega?



Introdução

22

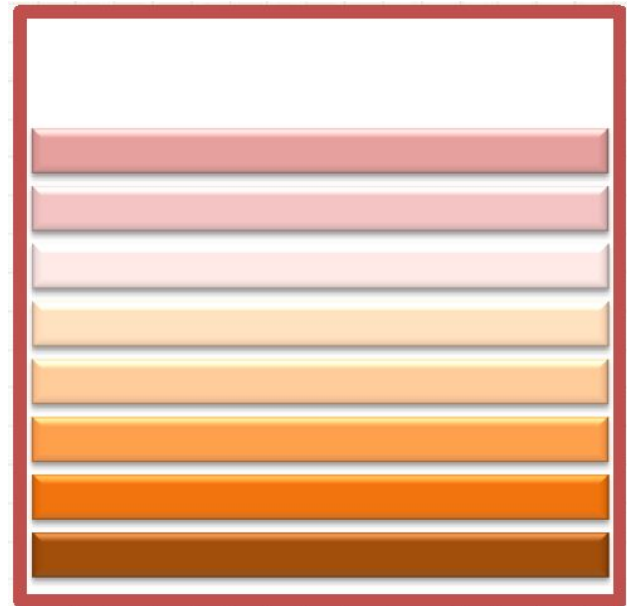
- Como os pedidos vão sendo **armazenados** para entrega?



Introdução

23

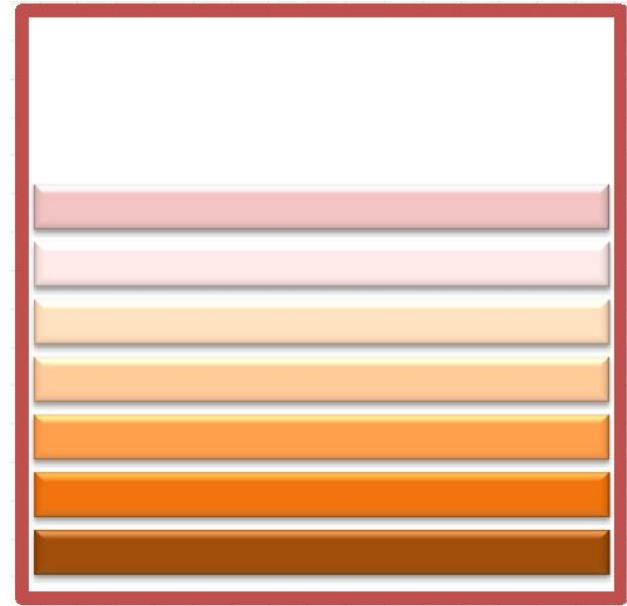
- E nas entregas?



Introdução

24

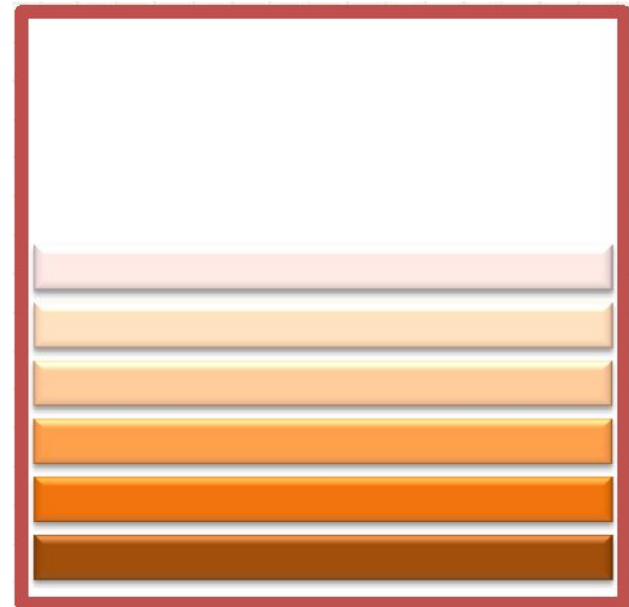
- E nas entregas?



Introdução

25

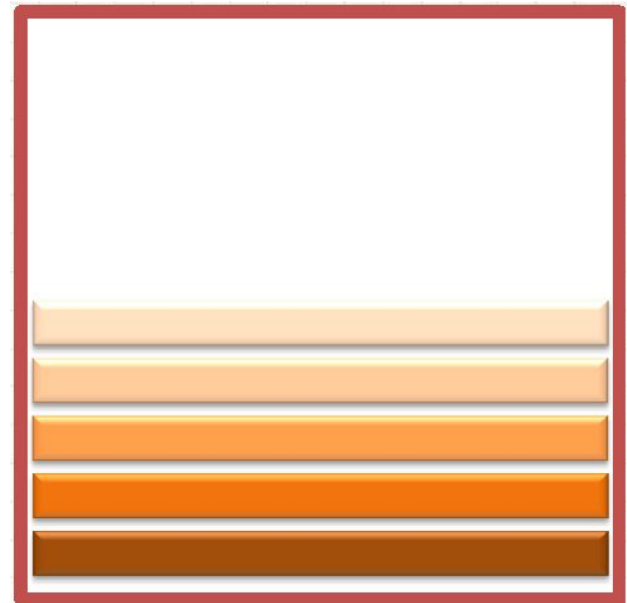
- E nas entregas?



Introdução

26

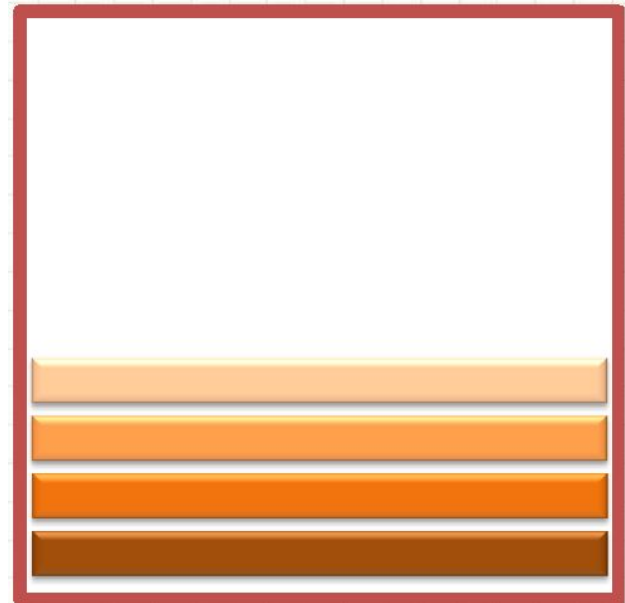
- E nas entregas?



Introdução

27

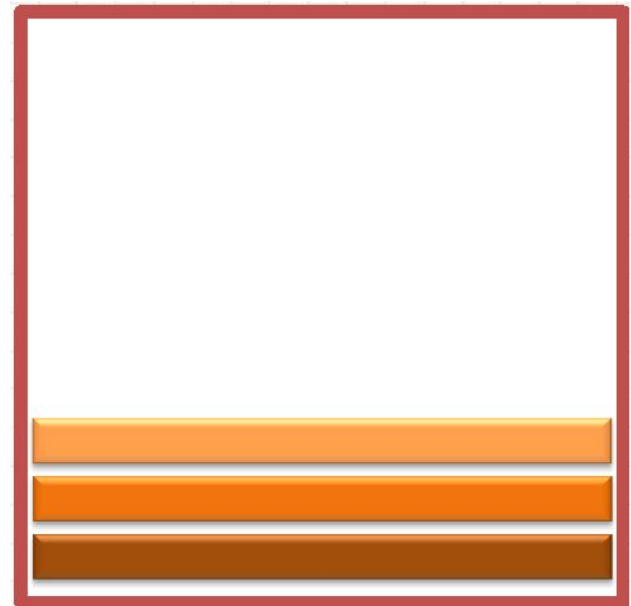
- E nas entregas?



Introdução

28

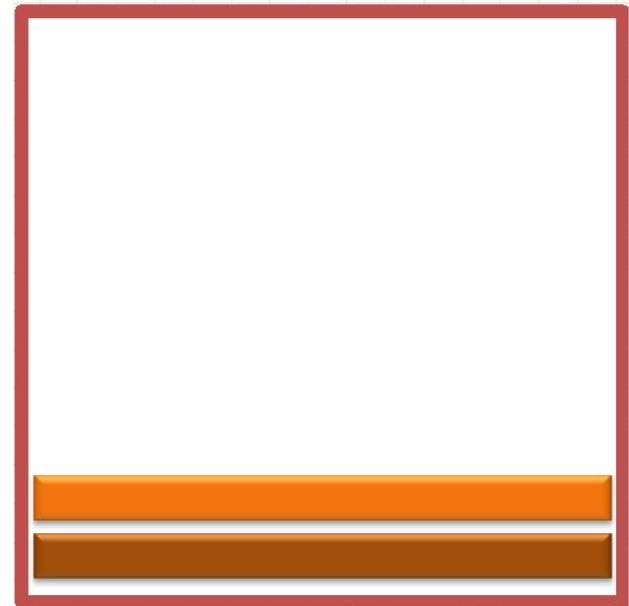
- E nas entregas?



Introdução

29

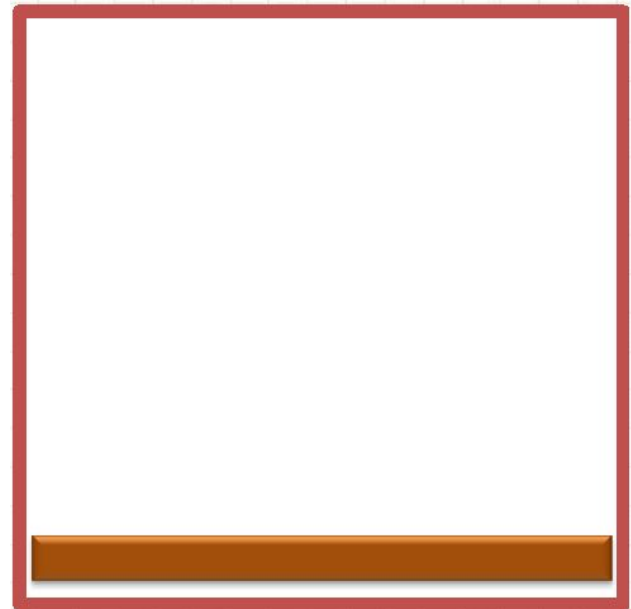
- E nas entregas?



Introdução

30

- E nas entregas?



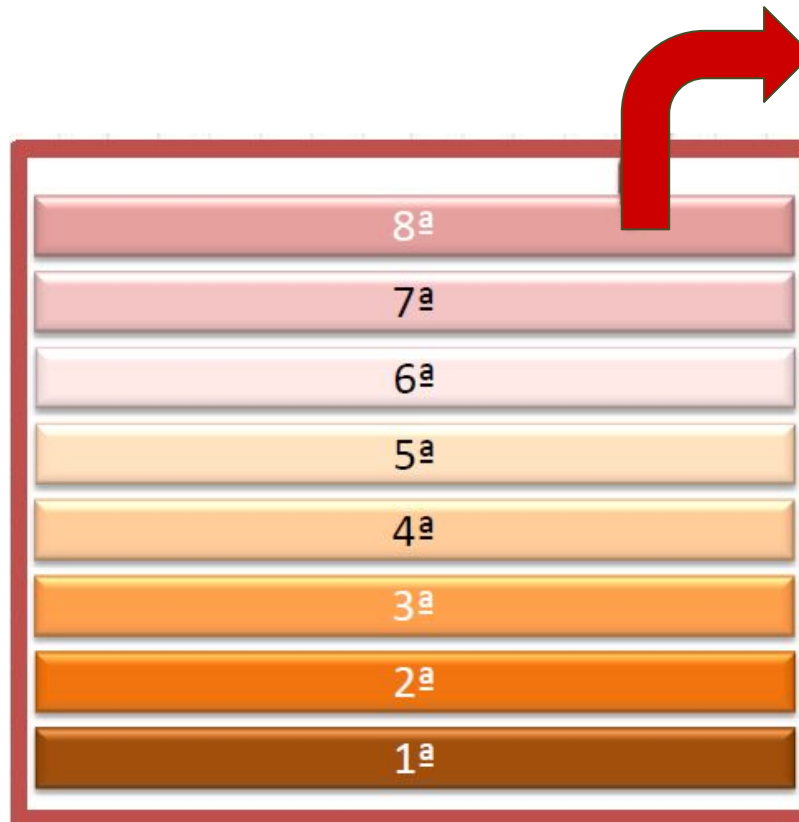
Introdução

31

❏ Resumindo

- A última pizza a entrar...
- Será a primeira a sair

LIFO:
Last in
First Out



Pilha de dados

Estrutura de pilha de dados

Aplicações de pilha de dados

Pilha de dados

33

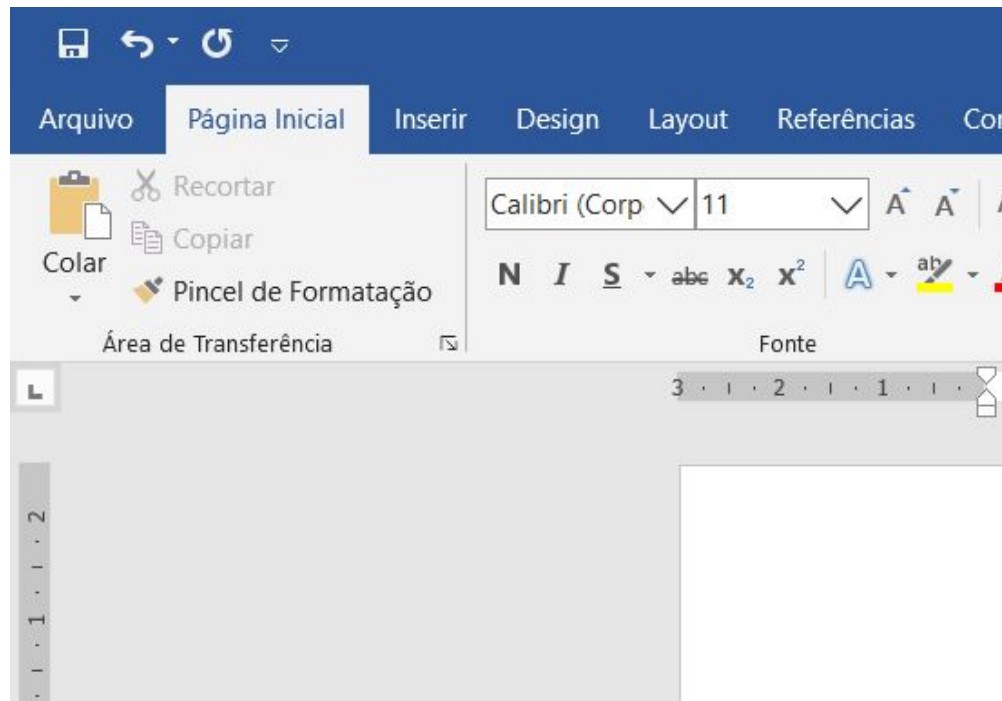
- ❑ Pilhas de dados
 - Estrutura de dados Pilha: Lista LIFO
 - Inserir: sempre no fim da lista (topo da pilha)
 - Remover: sempre no fim da lista (topo da pilha)

- ❑ Isso é útil em software?
 - Vejamos alguns casos!

Pilha de dados

34

- Já observou o recurso de “desfazer” do word?

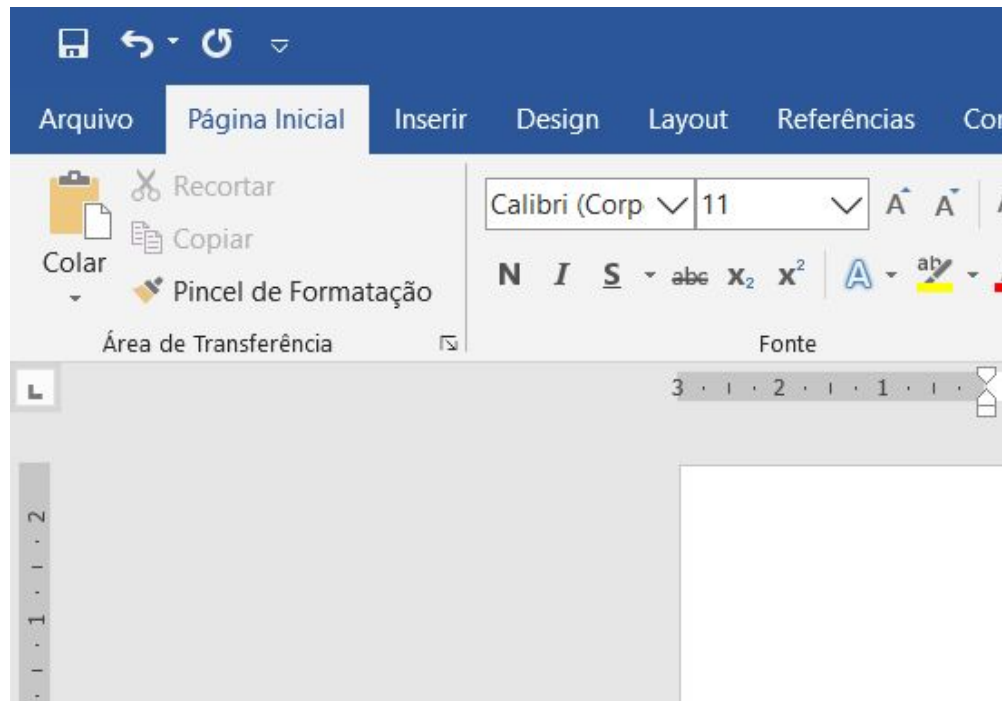


- Qual operação ele desfaz?

Pilha de dados

35

- ❑ Já observou o recurso de “desfazer” do word?

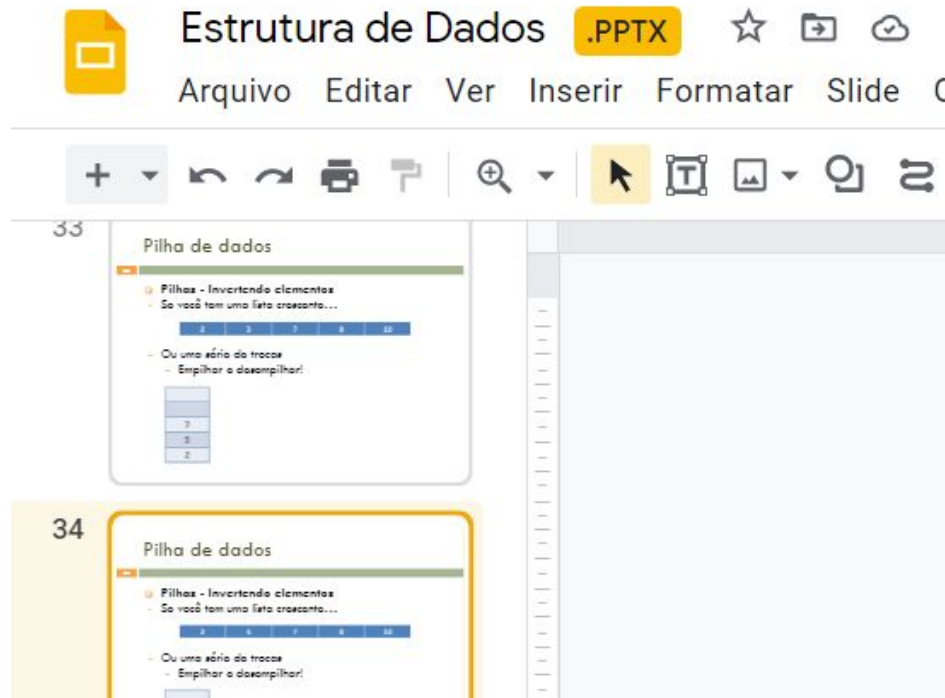


- ❑ Qual operação ele desfaz?
 - O word coloca as operações em uma pilha!

Pilha de dados

36

- ❑ Já observou o recurso de “desfazer” do powerpoint?



- ❑ Qual operação ele desfaz?
 - Powerpoint também coloca as operações em uma pilha!

Pilha de dados

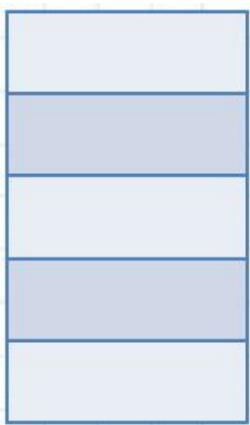
37

❑ Pilhas - Invertendo elementos

- Se você tem uma lista crescente...

| | | | | |
|---|---|---|---|----|
| 2 | 5 | 7 | 8 | 10 |
|---|---|---|---|----|

- Ou uma série de trocas
 - Empilhar e desempilhar!



Pilha de dados

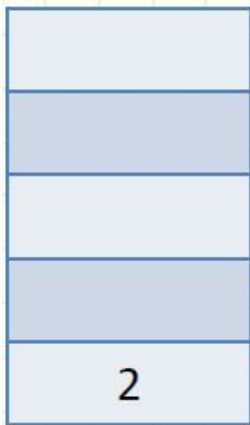
38

❑ Pilhas - Invertendo elementos

- Se você tem uma lista crescente...



- Ou uma série de trocas
 - Empilhar e desempilhar!



Pilha de dados

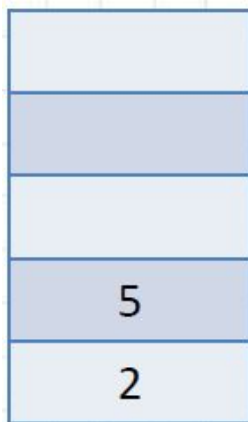
39

❑ Pilhas - Invertendo elementos

- Se você tem uma lista crescente...

| | | | | |
|---|---|---|---|----|
| 2 | 5 | 7 | 8 | 10 |
|---|---|---|---|----|

- Ou uma série de trocas
 - Empilhar e desempilhar!



Pilha de dados

40

❑ Pilhas - Invertendo elementos

- Se você tem uma lista crescente...

| | | | | |
|---|---|---|---|----|
| 2 | 5 | 7 | 8 | 10 |
|---|---|---|---|----|

- Ou uma série de trocas
 - Empilhar e desempilhar!

| |
|---|
| |
| |
| 7 |
| 5 |
| 2 |

Pilha de dados

41

❑ Pilhas - Invertendo elementos

- Se você tem uma lista crescente...

| | | | | |
|---|---|---|---|----|
| 2 | 5 | 7 | 8 | 10 |
|---|---|---|---|----|

- Ou uma série de trocas
 - Empilhar e desempilhar!

| |
|---|
| |
| 8 |
| 7 |
| 5 |
| 2 |

Pilha de dados

42

❑ Pilhas - Invertendo elementos

- Se você tem uma lista crescente...

| | | | | |
|---|---|---|---|----|
| 2 | 5 | 7 | 8 | 10 |
|---|---|---|---|----|

- Ou uma série de trocas
 - Empilhar e desempilhar!

| |
|----|
| 10 |
| 8 |
| 7 |
| 5 |
| 2 |

Pilha de dados

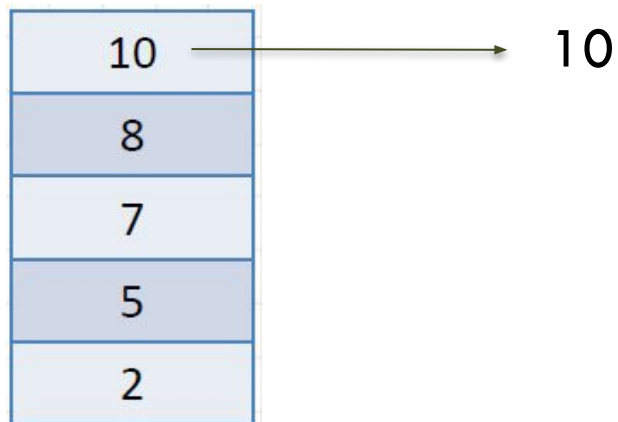
43

❑ Pilhas - Invertendo elementos

- Se você tem uma lista crescente...

| | | | | |
|---|---|---|---|----|
| 2 | 5 | 7 | 8 | 10 |
|---|---|---|---|----|

- Ou uma série de trocas
 - Empilhar e desempilhar!



Pilha de dados

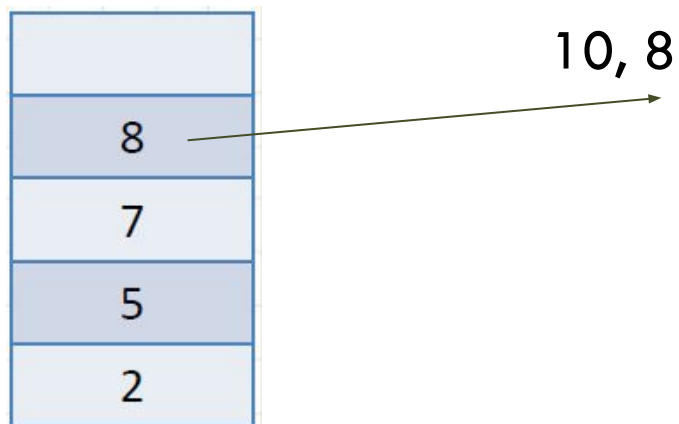
44

❑ Pilhas - Invertendo elementos

- Se você tem uma lista crescente...

| | | | | |
|---|---|---|---|----|
| 2 | 5 | 7 | 8 | 10 |
|---|---|---|---|----|

- Ou uma série de trocas
 - Empilhar e desempilhar!



Pilha de dados

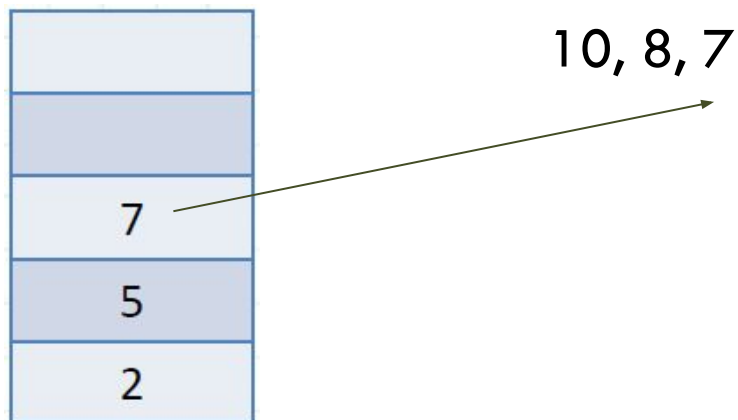
45

❑ Pilhas - Invertendo elementos

- Se você tem uma lista crescente...



- Ou uma série de trocas
 - Empilhar e desempilhar!



Pilha de dados

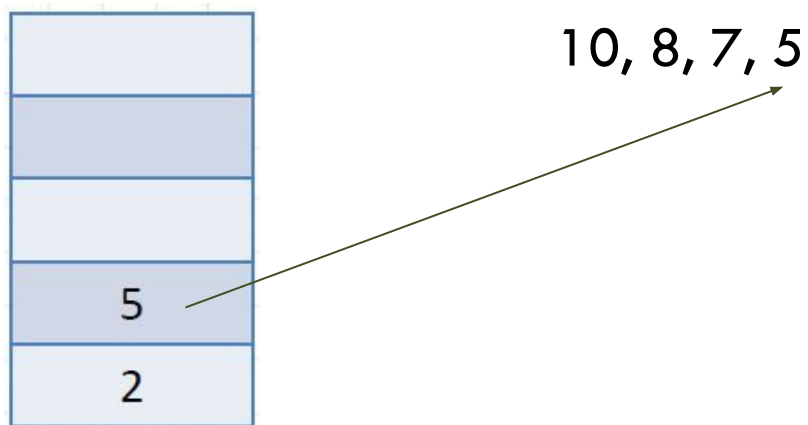
46

❑ Pilhas - Invertendo elementos

- Se você tem uma lista crescente...



- Ou uma série de trocas
 - Empilhar e desempilhar!



Pilha de dados

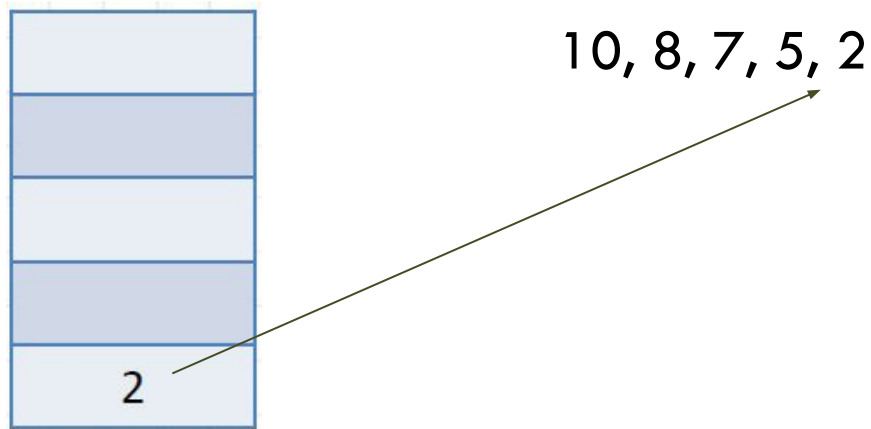
47

❑ Pilhas - Invertendo elementos

- Se você tem uma lista crescente...



- Ou uma série de trocas
 - Empilhar e desempilhar!

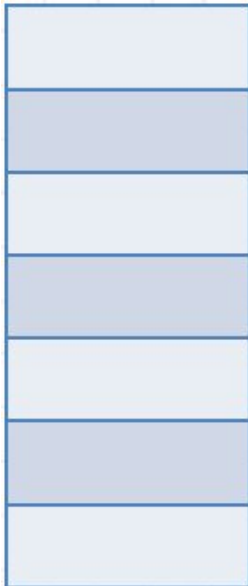


Pilha de dados

48

- ❑ **Pilhas - Fazendo cálculos**
 - Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$

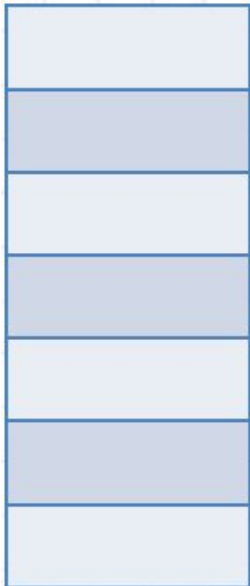


Pilha de dados

49

- ❑ **Pilhas - Fazendo cálculos**
 - Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



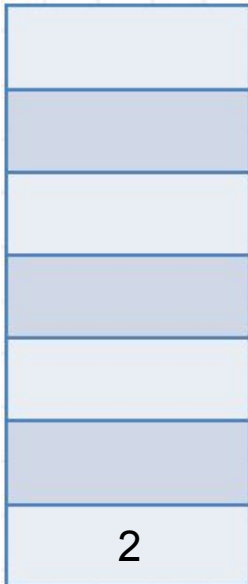
Pilha de dados

50

❑ Pilhas - Fazendo cálculos

- Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



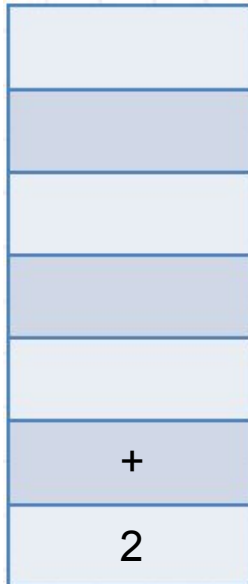
Pilha de dados

51

❏ Pilhas - Fazendo cálculos

- Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



Pilha de dados

52

❏ Pilhas - Fazendo cálculos

- Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



$$3 + 2 = 5$$

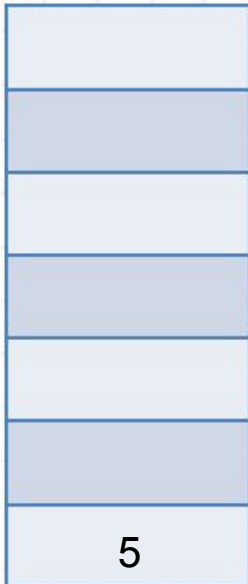
Pilha de dados

53

❑ Pilhas - Fazendo cálculos

- Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



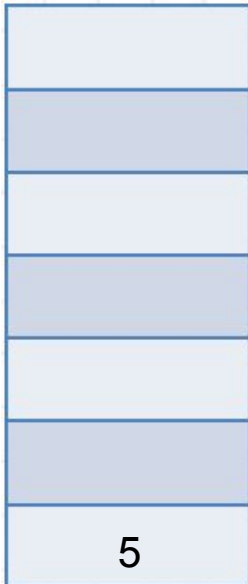
Pilha de dados

54

❑ Pilhas - Fazendo cálculos

- Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



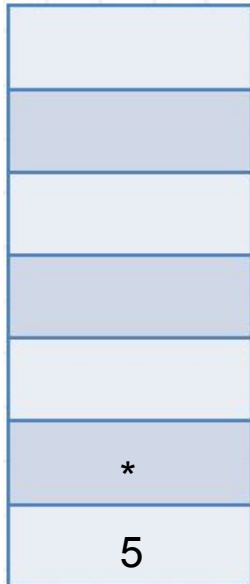
Pilha de dados

55

❑ Pilhas - Fazendo cálculos

- Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



Pilha de dados

56

❏ Pilhas - Fazendo cálculos

- Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



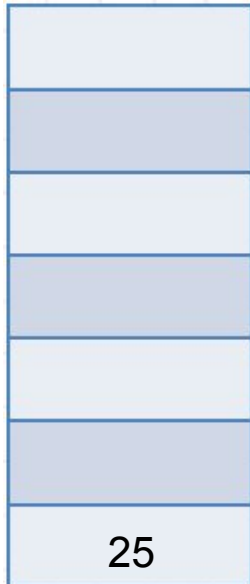
$$5 * 5 = 25$$

Pilha de dados

57

- ❑ **Pilhas - Fazendo cálculos**
 - Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$

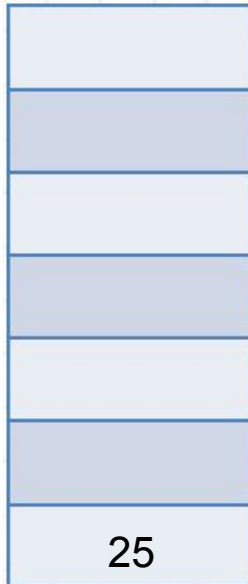


Pilha de dados

58

- ❑ **Pilhas - Fazendo cálculos**
 - Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



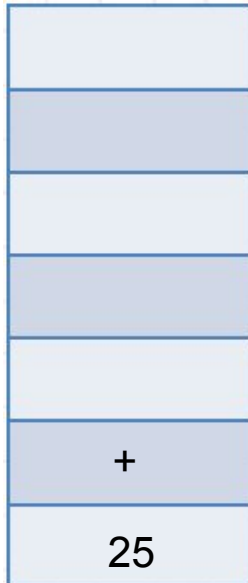
Pilha de dados

59

❑ Pilhas - Fazendo cálculos

- Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



Pilha de dados

60

- ❑ **Pilhas - Fazendo cálculos**
 - Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



Pilha de dados

61

- ❑ **Pilhas - Fazendo cálculos**
 - Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



Pilha de dados

62

❑ Pilhas - Fazendo cálculos

- Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$

| |
|----|
| |
| |
| 3 |
| / |
| 3 |
| + |
| 25 |

Pilha de dados

63

- ❑ **Pilhas - Fazendo cálculos**
 - Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$

| |
|----|
| |
| * |
| 3 |
| / |
| 3 |
| + |
| 25 |

Pilha de dados

64

❏ Pilhas - Fazendo cálculos

- Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$

| |
|----|
| 7 |
| * |
| 3 |
| / |
| 3 |
| + |
| 25 |

$$7 * 3 = 21$$

Pilha de dados

65

- ❑ **Pilhas - Fazendo cálculos**
 - Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$

| |
|----|
| |
| |
| 21 |
| / |
| 3 |
| + |
| 25 |

Pilha de dados

66

- ❑ **Pilhas - Fazendo cálculos**
 - Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$

| |
|----|
| |
| |
| 21 |
| / |
| 3 |
| + |
| 25 |

$$21 / 3 = 7$$

Pilha de dados

67

- ❑ **Pilhas - Fazendo cálculos**
 - Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



Pilha de dados

68

- ❑ **Pilhas - Fazendo cálculos**
 - Como fazemos esse cálculo?

$$(((2 + 3) * 5) + (3 / (3 * 7)))$$



$$7 + 25 = 32$$

70

Implementando uma pilha estática

Operações: inicializar, pilha cheia e vazia, empilhar e desempilhar

Pilha estática

71

- ❑ Na implementação de pilhas estáticas pode surgir várias formas de construção do tipo de dado pilha
 - Vetor e topo separados
 - Usando uma estrutura que conterá o vetor (armazena os elementos da pilha) e o topo

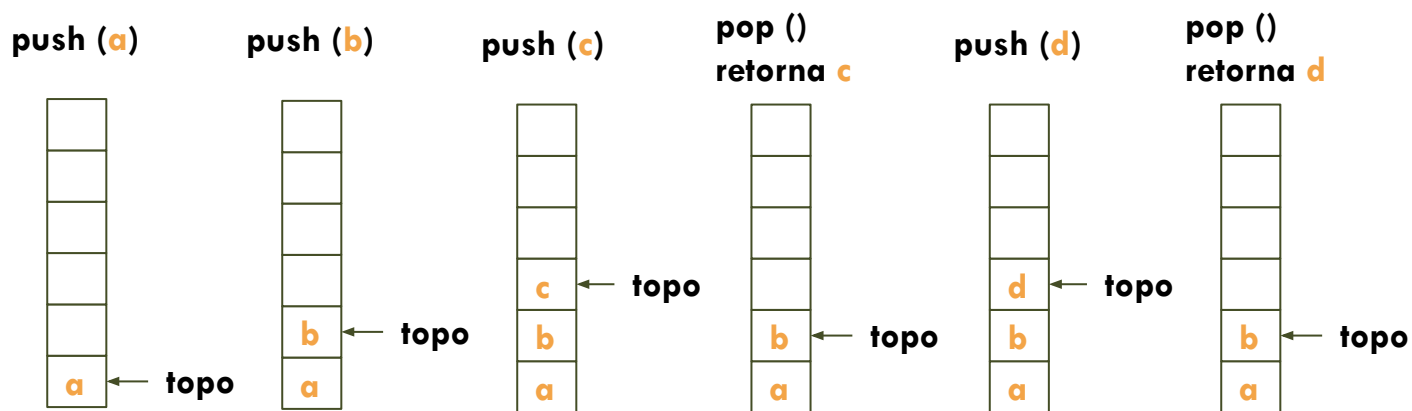
```
#define TAM 10
typedef struct pilha Pilha;

struct pilha{
    int info[TAM];
    int topo;
};
```

Pilha estática

72

- ❑ Novo elemento é inserido no topo e acesso é apenas ao topo
 - O primeiro que sai é o último que entrou (LIFO – last in, first out)



Pilha estática

73

- ❑ Novo elemento é inserido no topo e acesso é apenas ao topo
 - O primeiro que sai é o último que entrou (LIFO – last in, first out)

- ❑ Operações básicas:
 - **Inicializar:** recebe uma pilha e aponta seu topo para -1
 - **Empilhar (push):** insere um novo elemento no topo
 - **Desempilhar (pop):** remove um elemento do topo
 - **Pilha vazia:** verifica se pilha está vazia
 - **Pilha cheia:** verificar se pilha está cheia

Implementando uma pilha estática

74

❑ Elementos da pilha: inseridos em um vetor

Pilha:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |

Topo: ??

❑ Operações:

- Inicializar
- Pilha cheia e vazia
- Empilhar
- Desempilhar

Implementando uma pilha estática

75

❑ Inicializar uma pilha

Pilha:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |

Topo: -1



❑ Recebe uma pilha e aponta seu topo para -1

- Topo sempre indica o último elemento! Inicializar o topo com valor negativo indica que a pilha está vazia

Implementando uma pilha estática

76

❑ Empilhar

| | | | | | | | | | | |
|--------|---|---|---|---|---|---|---|---|---|---|
| Pilha: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |

Topo: -1

❑ Como empilhar um valor?

- Se $\text{topo} < (\text{TAM} - 1) \rightarrow$ Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 8?

Implementando uma pilha estática

77

Empilhar

Pilha:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| ? | ? | ? | ? | ? | ? | ? | ? | ? | ? |

Topo: -1

-1 < 9...
Posso empilhar!

Como empilhar um valor?

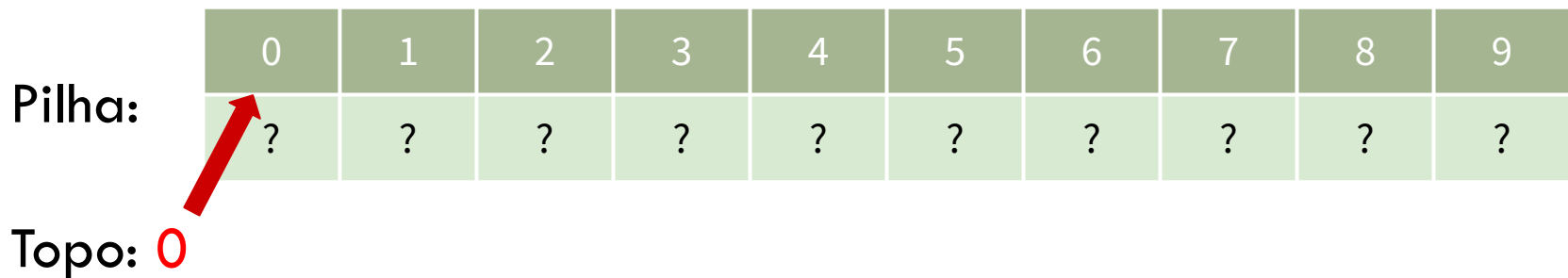
- Se $\text{topo} < (\text{TAM} - 1) \rightarrow$ Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

Vamos empilhar o número 8?

Implementando uma pilha estática

78

❑ Empilhar



❑ Como empilhar um valor?

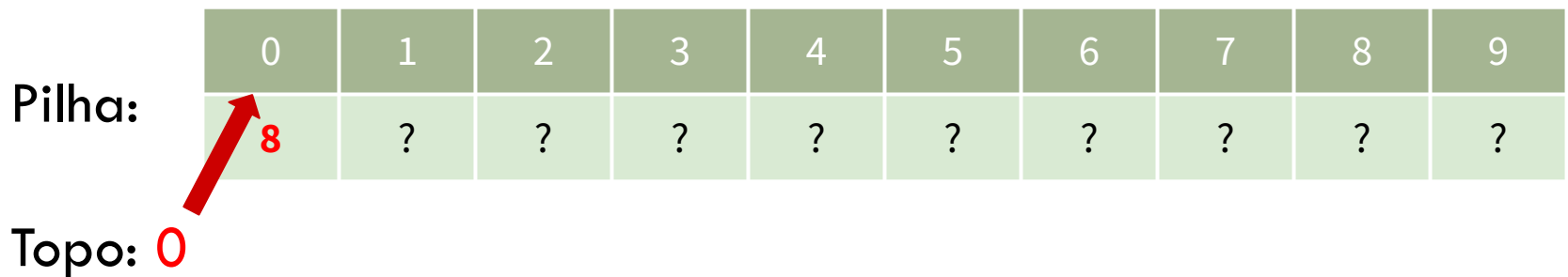
- Se $\text{topo} < (\text{TAM} - 1) \rightarrow$ Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 8?

Implementando uma pilha estática

79

❑ Empilhar



❑ Como empilhar um valor?

- Se $\text{topo} < (\text{TAM} - 1) \rightarrow$ Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 8?

Implementando uma pilha estática

80

❑ Empilhar

| | | | | | | | | | | |
|--------|---|---|---|---|---|---|---|---|---|---|
| Pilha: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 8 | ? | ? | ? | ? | ? | ? | ? | ? | ? |

Topo: 0

❑ Como empilhar um valor?

- Se $\text{topo} < (\text{TAM} - 1)$ → Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 5?

Implementando uma pilha estática

81

❑ Empilhar

Pilha:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 8 | ? | ? | ? | ? | ? | ? | ? | ? | ? |

Topo: 0

$0 < 9...$
Posso empilhar!

❑ Como empilhar um valor?

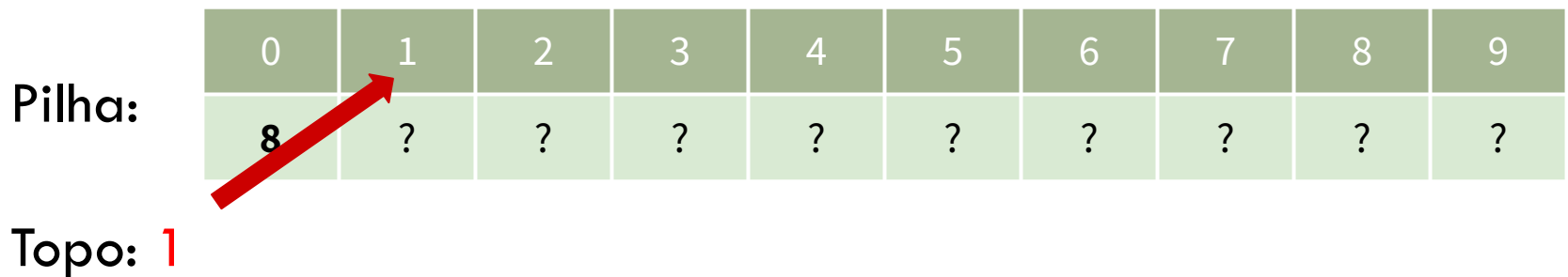
- Se $\text{topo} < (\text{TAM} - 1) \rightarrow$ Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 5?

Implementando uma pilha estática

82

❑ Empilhar



❑ Como empilhar um valor?

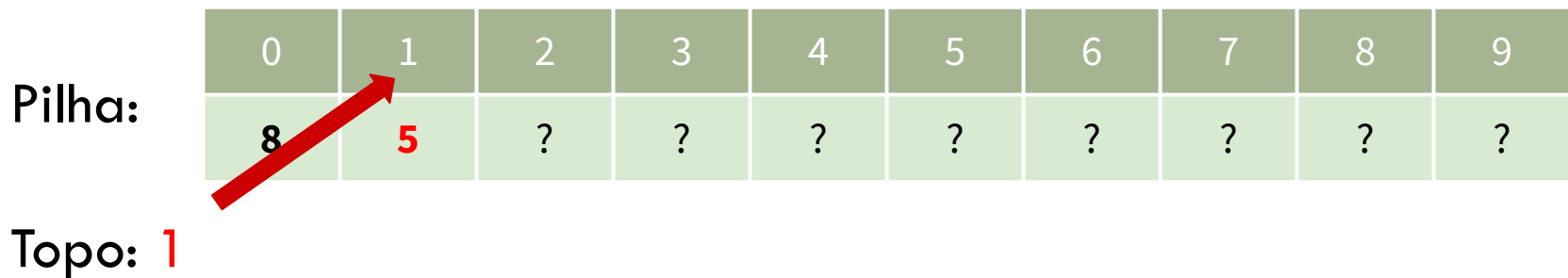
- Se $\text{topo} < (\text{TAM} - 1) \rightarrow$ Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 5?

Implementando uma pilha estática

83

❑ Empilhar



❑ Como empilhar um valor?

- Se $\text{topo} < (\text{TAM} - 1) \rightarrow$ Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 5?

Implementando uma pilha estática

84

❑ Empilhar

| | | | | | | | | | | |
|--------|---|---|----|----|----|----|----|----|----|---|
| Pilha: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 8 | 5 | 10 | 18 | 45 | 19 | 29 | 55 | 60 | ? |

Topo: 8

❑ Como empilhar um valor?

- Se $\text{topo} < (\text{TAM} - 1)$ → Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 78?

Implementando uma pilha estática

85

❑ Empilhar

Pilha:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|----|----|----|----|----|----|----|---|
| 8 | 5 | 10 | 18 | 45 | 19 | 29 | 55 | 60 | ? |

Topo: 8

8 < 9...
Posso empilhar!

❑ Como empilhar um valor?

- Se $\text{topo} < (\text{TAM} - 1) \rightarrow$ Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 78?

Implementando uma pilha estática

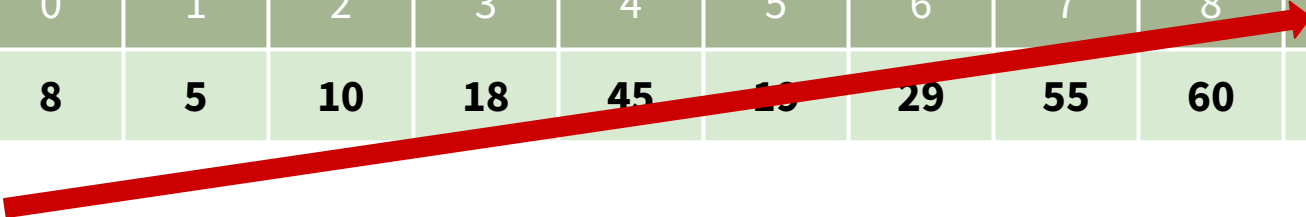
86

❑ Empilhar

Pilha:

| | | | | | | | | | |
|---|---|----|----|----|----|----|----|----|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 5 | 10 | 18 | 45 | 19 | 29 | 55 | 60 | ? |

Topo: 9



❑ Como empilhar um valor?

- Se $\text{topo} < (\text{TAM} - 1) \rightarrow$ Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número **78**?

Implementando uma pilha estática

87

❑ Empilhar

| | | | | | | | | | | |
|--------|---|---|----|----|----|----|----|----|----|----|
| Pilha: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 8 | 5 | 10 | 18 | 45 | 19 | 29 | 55 | 60 | 78 |

Topo: 9

❑ Como empilhar um valor?

- Se $\text{topo} < (\text{TAM} - 1)$ → Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 78?

Implementando uma pilha estática

88

❑ Empilhar

| | | | | | | | | | | |
|--------|---|---|----|----|----|----|----|----|----|----|
| Pilha: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 8 | 5 | 10 | 18 | 45 | 19 | 29 | 55 | 60 | 78 |

Topo: 9

❑ Como empilhar um valor?

- Se $\text{topo} < (\text{TAM} - 1)$ → Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 20?

Implementando uma pilha estática

89

❑ Empilhar

Pilha:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|----|----|----|----|----|----|----|----|
| 8 | 5 | 10 | 18 | 45 | 19 | 29 | 55 | 60 | 78 |

Topo: 9

9 < 9? Não!

❑ Como empilhar um valor?

- Se $\text{topo} < (\text{TAM} - 1) \rightarrow$ Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 20?

Implementando uma pilha estática

90

❑ Empilhar

Pilha:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|----|----|----|----|----|----|----|----|
| 8 | 5 | 10 | 18 | 45 | 19 | 29 | 55 | 60 | 78 |

Topo: 9

Pilha cheia!

❑ Como empilhar um valor?

- Se $\text{topo} < (\text{TAM} - 1) \rightarrow$ Pode empilhar
- Soma 1 no topo... E acrescenta-se elemento na posição do vetor

❑ Vamos empilhar o número 20?

Implementando uma pilha estática

91

❑ Inicializar

- Recebe uma pilha e aponta seu topo para -1

```
void pilha_inicializa (Pilha* p){  
    p->topo = -1;  
}
```

Implementando uma pilha estática

92

❑ Pilha cheia

- Verifica se pilha está cheia, caso esteja, não pode empilhar

```
int pilha_cheia(Pilha* p){  
    if(p->topo >= TAM - 1)  
        return 1;  
    return 0;  
}
```



Pilha cheia!

Implementando uma pilha estática

93

❑ Empilhar (push)

- Se a pilha não estiver cheia, incrementa o topo e insere um novo elemento na posição do topo atual

```
int pilha_push(Pilha* p, int valor){  
    if(!pilha_cheia(p)){  
        p->topo++;  
        p->info[p->topo] = valor;  
        return 1;  
    }  
    return 0;  
}
```

Implementando uma pilha estática

94

❑ Desempilhar

| | | | | | | | | | | |
|--------|---|---|----|----|----|----|---|---|---|---|
| Pilha: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 8 | 5 | 10 | 18 | 45 | 19 | ? | ? | ? | ? |

Topo: 5

- ❑ Como desempilhar um valor?
 - Se o topo $\neq -1$ → Pode desempilhar
 - Lê o elemento... E subtrai 1 do topo
- ❑ Vamos desempilhar um número?

Implementando uma pilha estática

95

❑ Desempilhar

Pilha:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|----|----|----|----|---|---|---|---|
| 8 | 5 | 10 | 18 | 45 | 19 | ? | ? | ? | ? |

Topo: 5

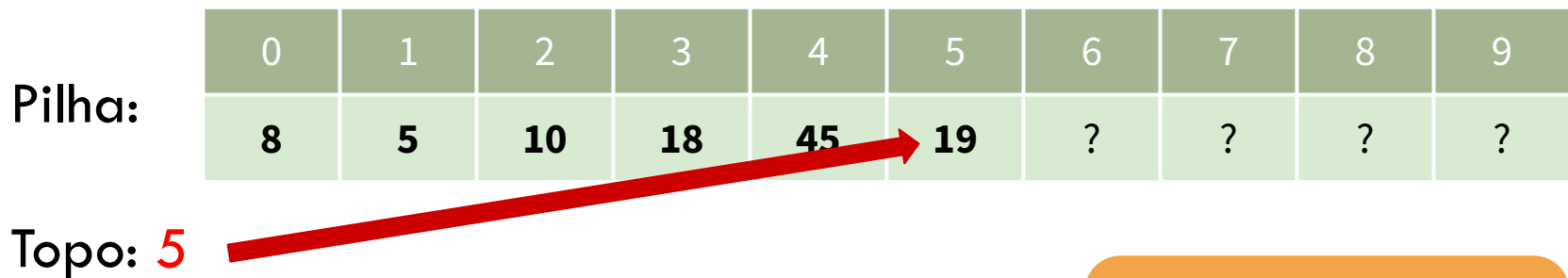
- ❑ Como desempilhar um valor?
 - Se o topo $\neq -1 \rightarrow$ Pode desempilhar
 - Lê o elemento... E subtrai 1 do topo
- ❑ Vamos desempilhar um número?

5 $\neq -1 \dots$
Posso desempilhar!

Implementando uma pilha estática

96

❑ Desempilhar



Desempilhamos:
19!

- ❑ Como desempilhar um valor?
 - Se o topo $\neq -1$ → Pode desempilhar
 - Lê o elemento... E subtrai 1 do topo
- ❑ Vamos desempilhar um número?

Implementando uma pilha estática

97

❑ Desempilhar

| | | | | | | | | | | |
|--------|---|---|----|----|----|----|---|---|---|---|
| Pilha: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 8 | 5 | 10 | 18 | 45 | 19 | ? | ? | ? | ? |

Topo: 4

- ❑ Como desempilhar um valor?
 - Se o topo $\neq -1$ → Pode desempilhar
 - Lê o elemento... E subtrai 1 do topo
- ❑ Vamos desempilhar outro número?

Implementando uma pilha estática

98

❑ Desempilhar

Pilha:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|----|----|----|----|---|---|---|---|
| 8 | 5 | 10 | 18 | 45 | 19 | ? | ? | ? | ? |

Topo: 4

4 \neq -1 ...
Posso desempilhar!

- ❑ Como desempilhar um valor?
 - Se o topo \neq -1 → Pode desempilhar
 - Lê o elemento... E subtrai 1 do topo
- ❑ Vamos desempilhar outro número?

Implementando uma pilha estática

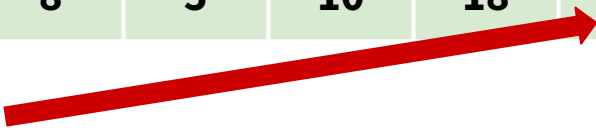
99

❑ Desempilhar

Pilha:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|----|----|----|----|---|---|---|---|
| 8 | 5 | 10 | 18 | 45 | 19 | ? | ? | ? | ? |

Topo: 4



Desempilhamos:
45!

- ❑ Como desempilhar um valor?
 - Se o topo $\neq -1$ → Pode desempilhar
 - Lê o elemento... E subtrai 1 do topo
- ❑ Vamos desempilhar outro número?

Implementando uma pilha estática

100

❑ Desempilhar

| | | | | | | | | | | |
|--------|---|---|----|----|----|----|---|---|---|---|
| Pilha: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 8 | 5 | 10 | 18 | 45 | 19 | ? | ? | ? | ? |

Topo: 3

- ❑ Como desempilhar um valor?
 - Se o topo $\neq -1$ → Pode desempilhar
 - Lê o elemento... E subtrai 1 do topo
- ❑ Vamos desempilhar outro número?

Implementando uma pilha estática

101

❑ Desempilhar

| | | | | | | | | | | |
|--------|---|---|----|----|----|----|---|---|---|---|
| Pilha: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 8 | 5 | 10 | 18 | 45 | 19 | ? | ? | ? | ? |

Topo: -1

- ❑ Como desempilhar um valor?
 - Se o topo $\neq -1 \rightarrow$ Pode desempilhar
 - Lê o elemento... E subtrai 1 do topo
- ❑ Vamos desempilhar outro número?

Implementando uma pilha estática

102

❑ Desempilhar

| | | | | | | | | | | |
|--------|---|---|----|----|----|----|---|---|---|---|
| Pilha: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 8 | 5 | 10 | 18 | 45 | 19 | ? | ? | ? | ? |

Topo: **-1**

-1 != -1? Não!

- ❑ Como desempilhar um valor?
 - Se o topo != -1 → Pode desempilhar
 - Lê o elemento... E subtrai 1 do topo
- ❑ Vamos desempilhar outro número?

Implementando uma pilha estática

103

❑ Desempilhar

| | | | | | | | | | | |
|--------|---|---|----|----|----|----|---|---|---|---|
| Pilha: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | 8 | 5 | 10 | 18 | 45 | 19 | ? | ? | ? | ? |

Topo: -1

Pilha vazia!

- ❑ Como desempilhar um valor?
 - Se o topo $\neq -1$ → Pode desempilhar
 - Lê o elemento... E subtrai 1 do topo
- ❑ Vamos desempilhar outro número?

Implementando uma pilha estática

104

❑ Pilha vazia

- Verifica se pilha está vazia, caso esteja, não pode desempilhar

```
int pilha_vazia(Pilha* p){  
    if(p -> topo == -1)  
        return 1;  
    return 0;  
}
```



Pilha vazia!

Implementando uma pilha estática

105

❑ Desempilhar (pop)

- Se a pilha não estiver vazia, lê o elemento e subtrai 1 do topo

```
int pilha_pop(Pilha *p){
    int valor = 0;
    if(!pilha_vazia(p)){
        valor = p->info[p->topo];
        p->topo--;
    }else{
        exit(1);
    }
    return valor;
}
```

Implementando uma pilha estática

Atividade

106

❑ Imprimir?

- Imprimir os elementos da pilha sem usar o `pilha_pop`, pois o objetivo é só observar o conteúdo pilha sem modificar o seu conteúdo

Implementando uma pilha dinâmica

Operações: inicializar, pilha cheia e vazia, empilhar e desempilhar

Pilha dinâmica

108

- ❑ Implementar uma pilha dinâmica usando uma lista encadeada simples

```
typedef struct pilhaD PilhaD;  
  
struct pilhaD{  
    int info;  
    PilhaD* prox;  
};
```

Pilha dinâmica

109

- ❑ Novo elemento é inserido no topo e acesso é apenas ao topo
 - O primeiro que sai é o último que entrou (LIFO – last in, first out)

- ❑ Operações básicas:
 - **Inicializar:** cria uma pilha vazia, representada pelo ponteiro NULL
 - **AlocaNo:** aloca memória para armazenar o elemento (nó)
 - **Empilhar (push):** insere um novo elemento no topo
 - **Desempilhar (pop):** remove um elemento do topo
 - **Pilha vazia:** verifica se pilha está vazia

Implementando uma pilha dinâmica

110

❏ Inicializar

- Cria uma pilha vazia, representada pelo ponteiro NULL

```
PilhaD* pilhaD_inicializa (){  
    return NULL;  
}
```

Pilha → NULL

Implementando uma pilha dinâmica

111

❏ Empilhar (push)

- Aloca memória para armazenar o elemento
- Encadeia o elemento na pilha existente

```
PilhaD* alocaNo(int valor){  
    PilhaD* no = (PilhaD*) malloc(sizeof(PilhaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
PilhaD* pilhaD_push(PilhaD* p, int valor){  
    PilhaD *novo = alocaNo(valor);  
    novo->prox = p;  
    return novo;  
}
```

Implementando uma pilha dinâmica

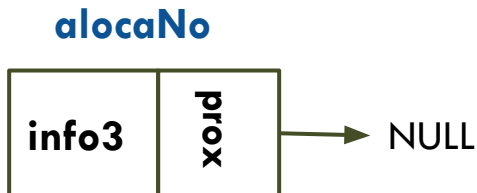
112

❏ Empilhar (push)

- Aloca memória para armazenar o elemento
- Encadeia o elemento na pilha existente

```
PilhaD* alocaNo(int valor){  
    PilhaD* no = (PilhaD*) malloc(sizeof(PilhaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
PilhaD* pilhaD_push(PilhaD* p, int valor){  
    PilhaD *novo = alocaNo(valor);  
    novo->prox = p;  
    return novo;  
}
```



Implementando uma pilha dinâmica

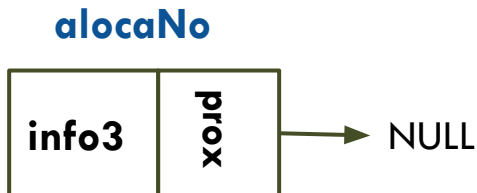
113

❏ Empilhar (push)

- Aloca memória para armazenar o elemento
- Encadeia o elemento na pilha existente

```
PilhaD* alocaNo(int valor){  
    PilhaD* no = (PilhaD*) malloc(sizeof(PilhaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
PilhaD* pilhaD_push(PilhaD* p, int valor){  
    PilhaD *novo = alocaNo(valor);  
    novo->prox = p;  
    return novo;  
}
```



Implementando uma pilha dinâmica

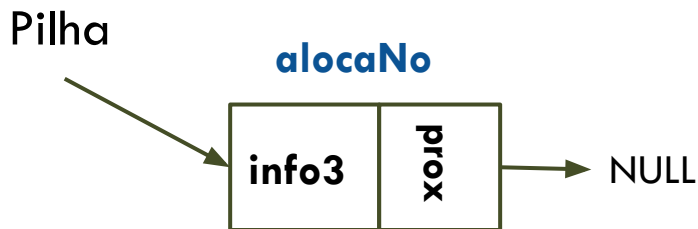
114

❏ Empilhar (push)

- Aloca memória para armazenar o elemento
- Encadeia o elemento na pilha existente

```
PilhaD* alocaNo(int valor){  
    PilhaD* no = (PilhaD*) malloc(sizeof(PilhaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
PilhaD* pilhaD_push(PilhaD* p, int valor){  
    PilhaD *novo = alocaNo(valor);  
    novo->prox = p;  
    return novo;  
}
```



Implementando uma pilha dinâmica

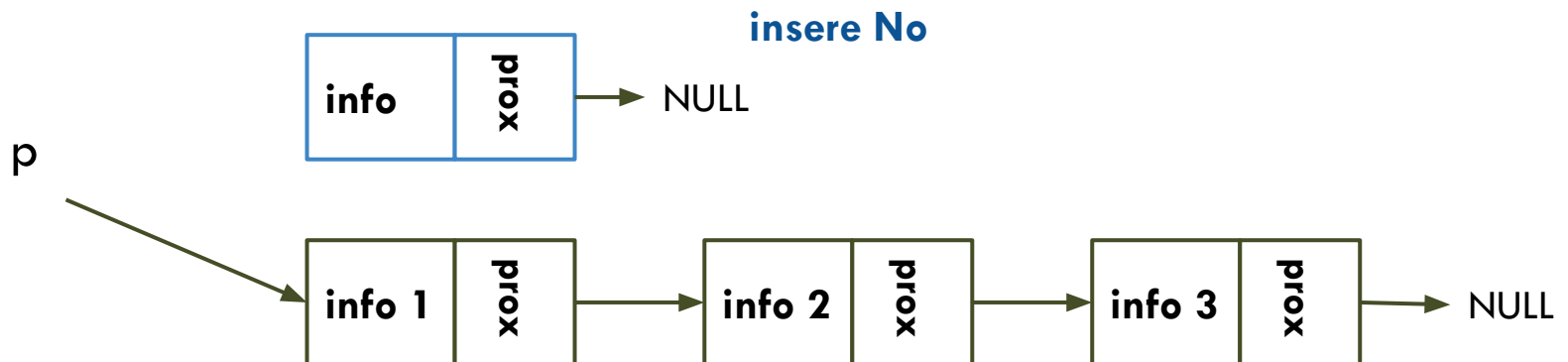
115

❏ Empilhar (push)

- Aloca memória para armazenar o elemento
- Encadeia o elemento na pilha existente

```
PilhaD* alocaNo(int valor){  
    PilhaD* no = (PilhaD*) malloc(sizeof(PilhaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
PilhaD* pilhaD_push(PilhaD* p, int valor){  
    PilhaD *novo = alocaNo(valor);  
    novo->prox = p;  
    return novo;  
}
```



Implementando uma pilha dinâmica

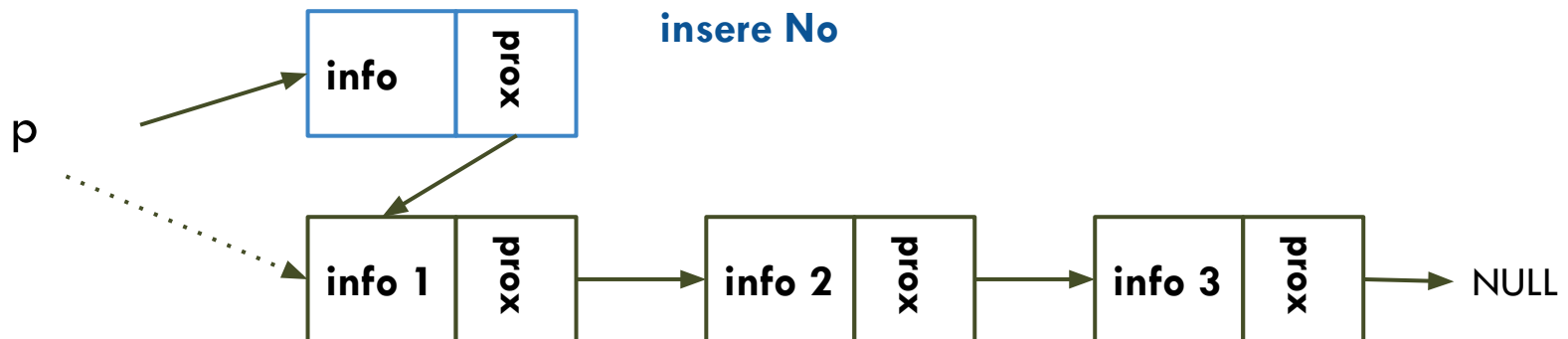
116

❏ Empilhar (push)

- Aloca memória para armazenar o elemento
- Encadeia o elemento na pilha existente

```
PilhaD* alocaNo(int valor){  
    PilhaD* no = (PilhaD*) malloc(sizeof(PilhaD));  
    no->info = valor;  
    no->prox = NULL;  
    return no;  
}
```

```
PilhaD* pilhaD_push(PilhaD* p, int valor){  
    PilhaD *novo = alocaNo(valor);  
    novo->prox = p;  
    return novo;  
}
```



Implementando uma pilha dinâmica

117

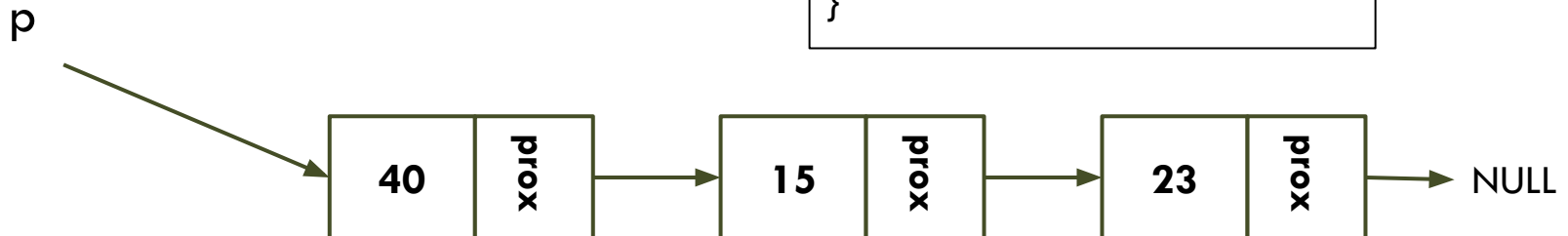
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

118

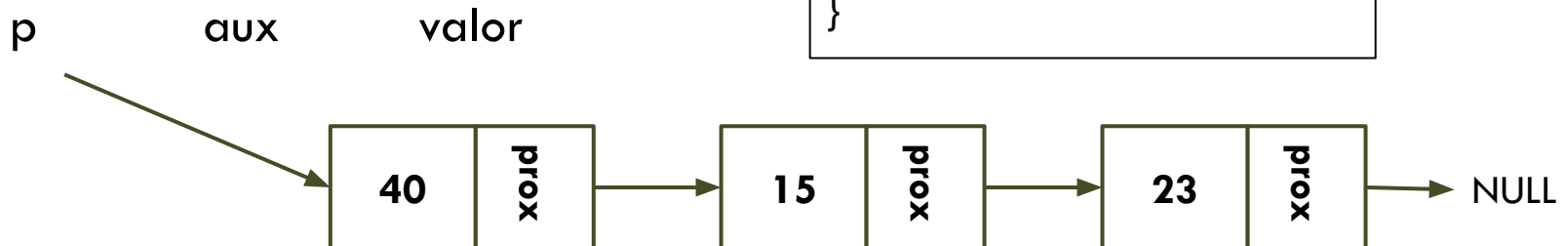
❏ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

119

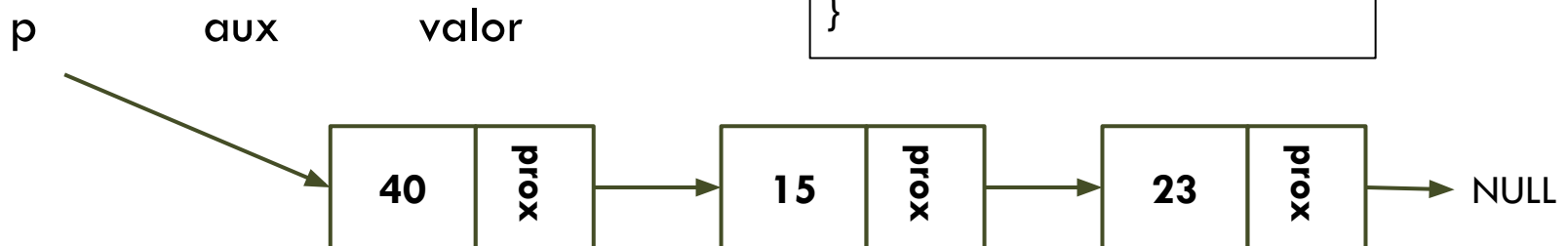
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

120

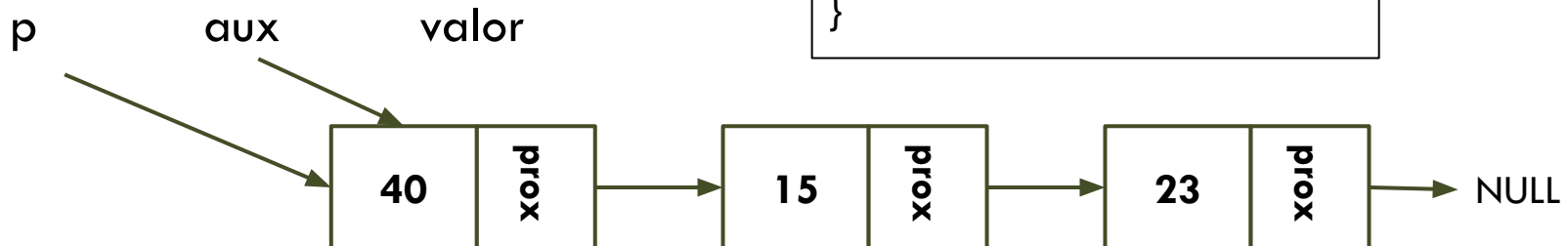
❏ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

121

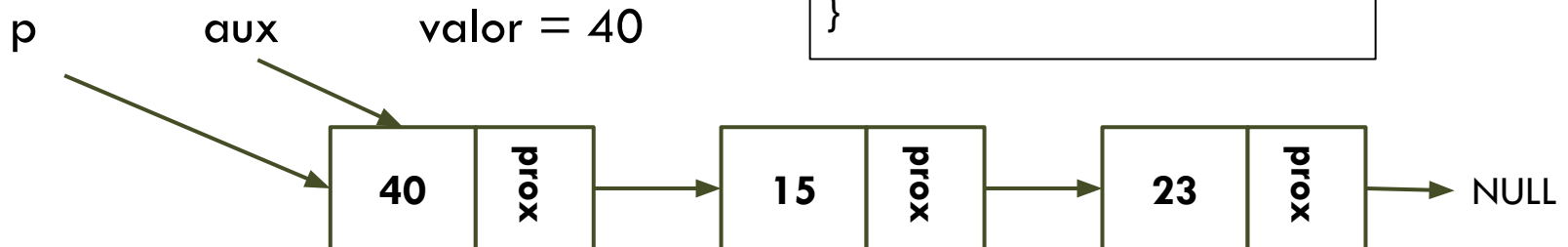
❏ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

122

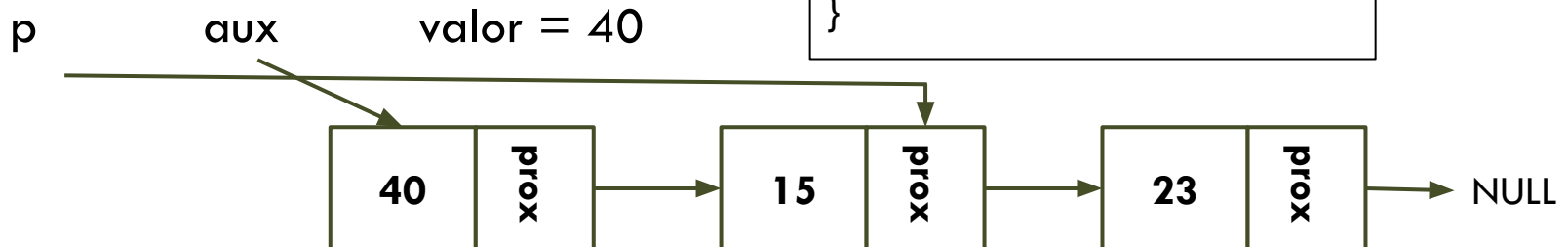
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

123

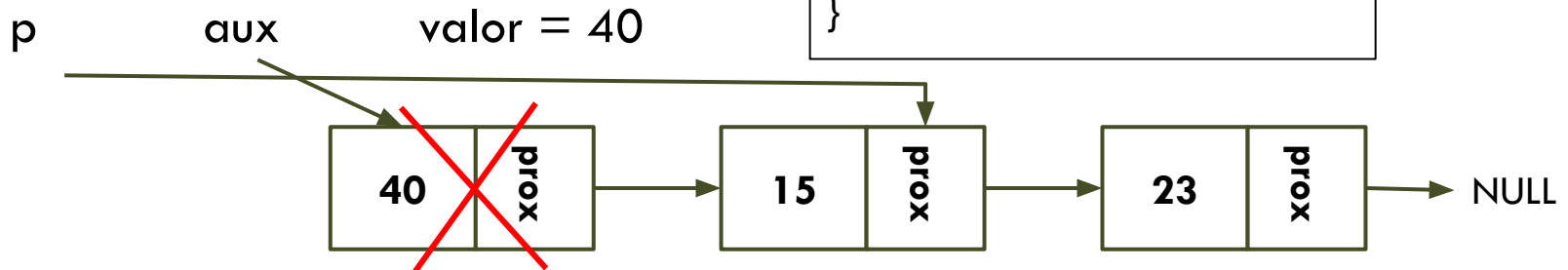
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

124

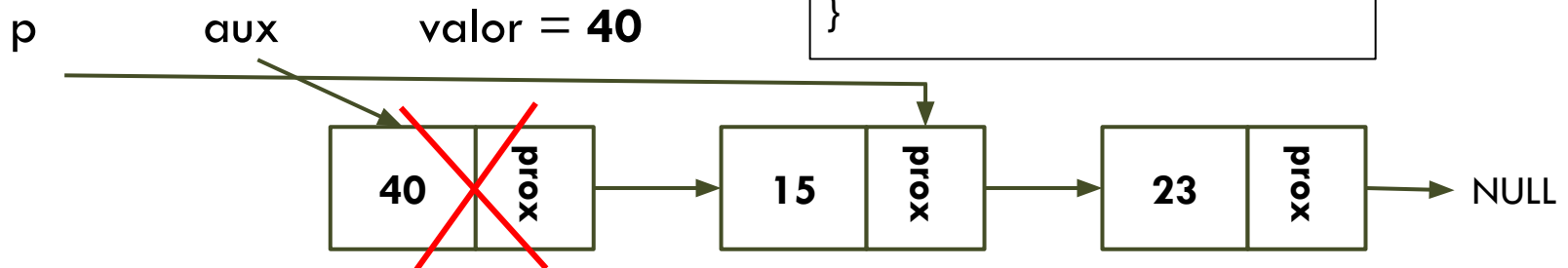
❏ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

125

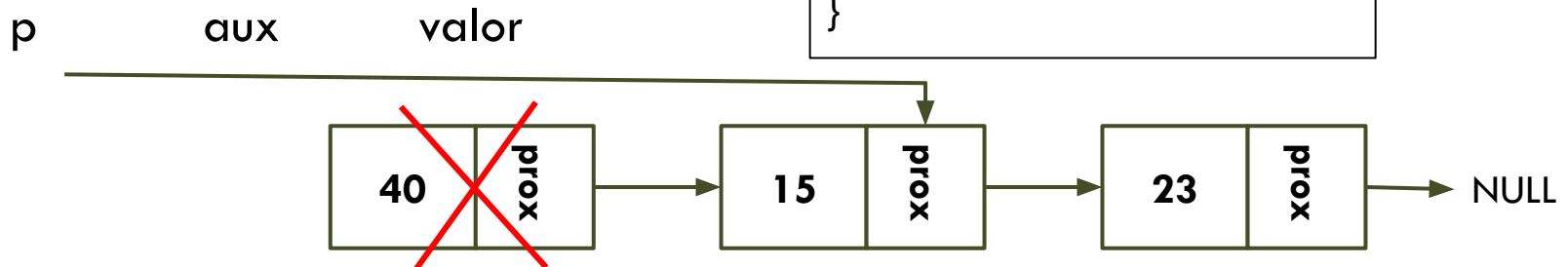
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

126

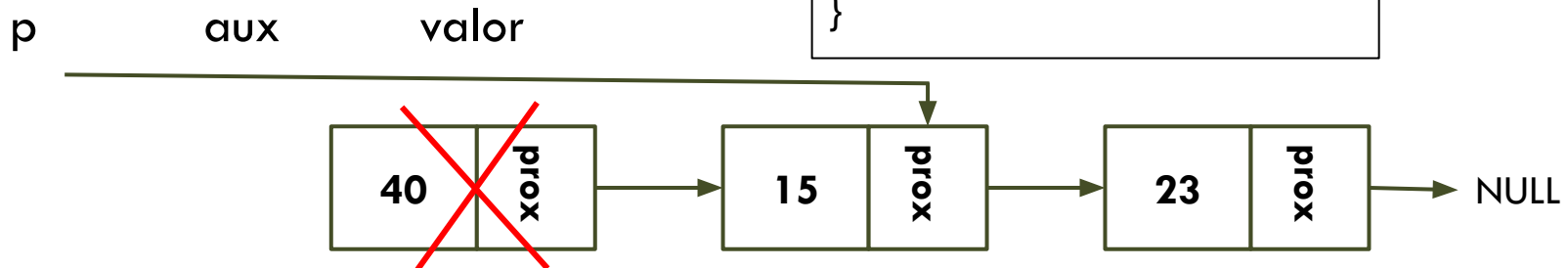
❏ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

127

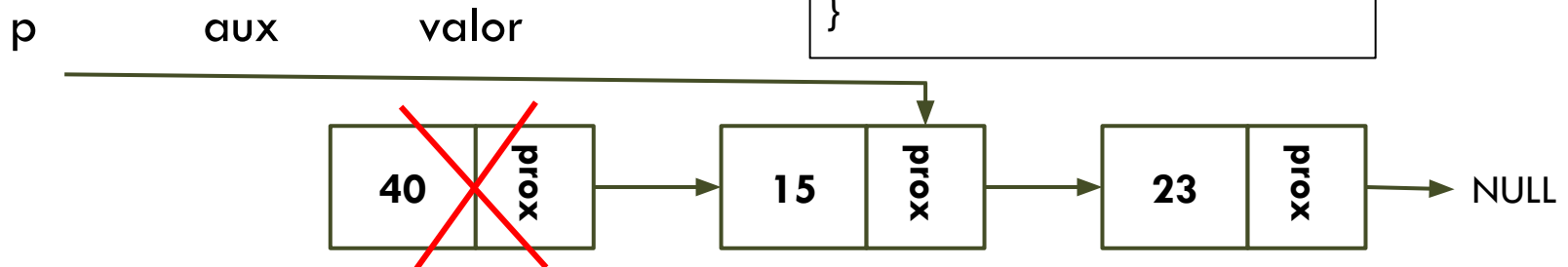
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

128

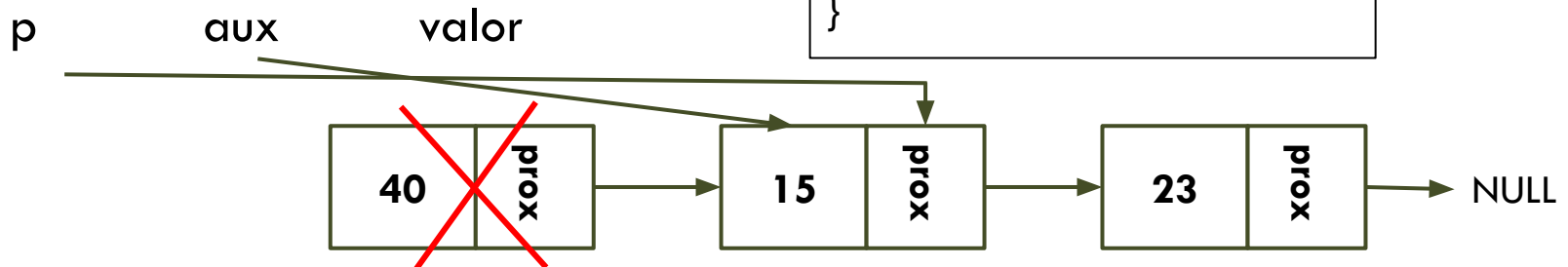
❏ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

129

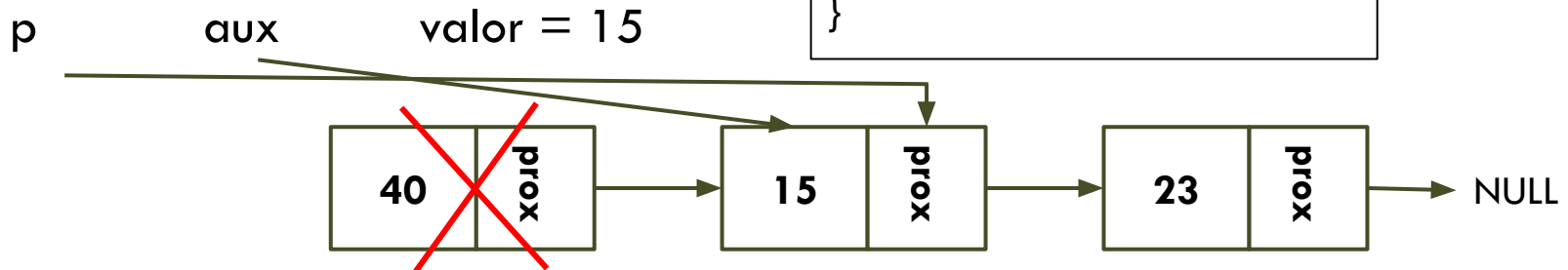
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

130

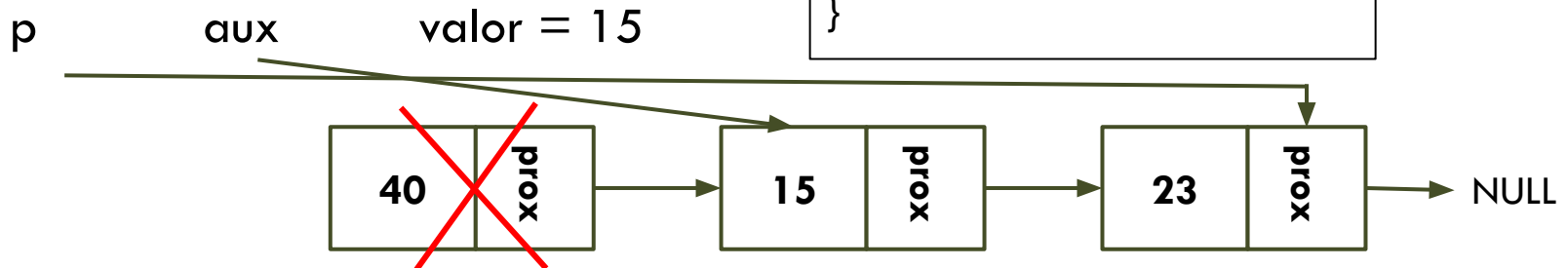
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

131

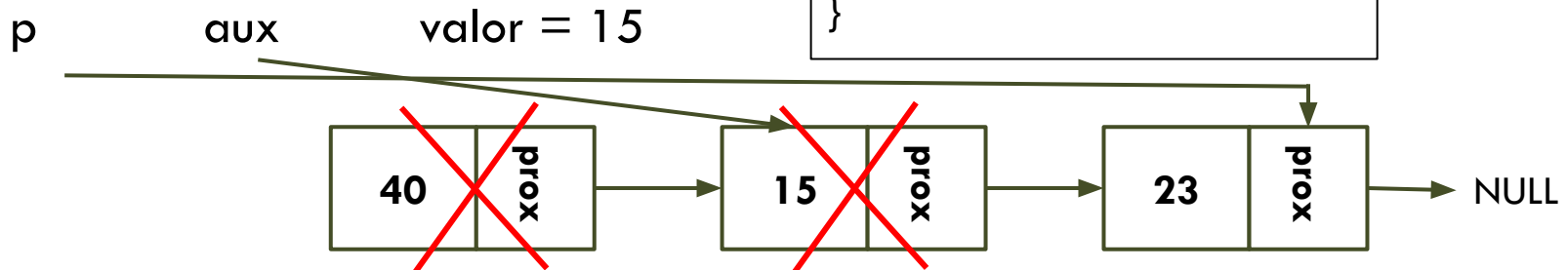
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

pilhaD_pop(&p);
pilhaD_pop(&p);
pilhaD_pop(&p);
pilhaD_pop(&p);



Implementando uma pilha dinâmica

132

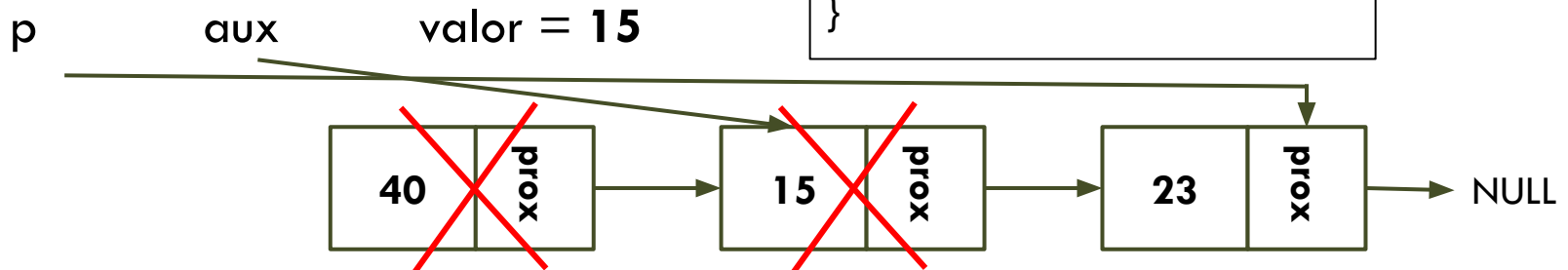
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

133

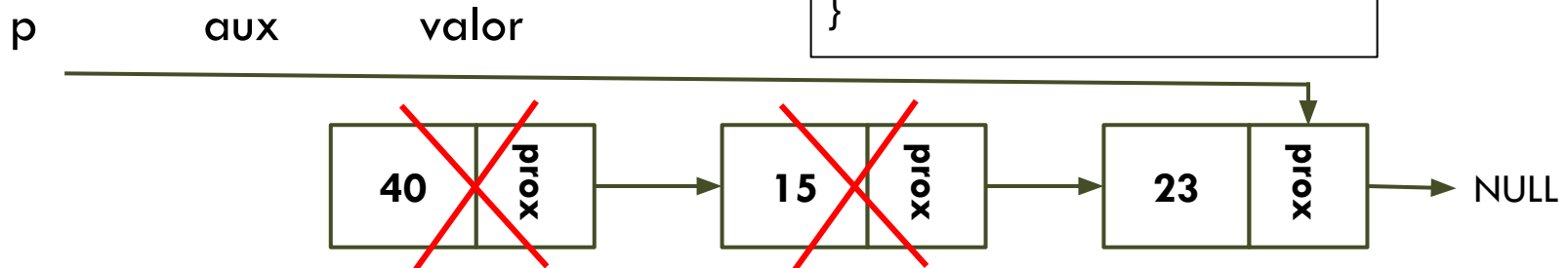
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

134

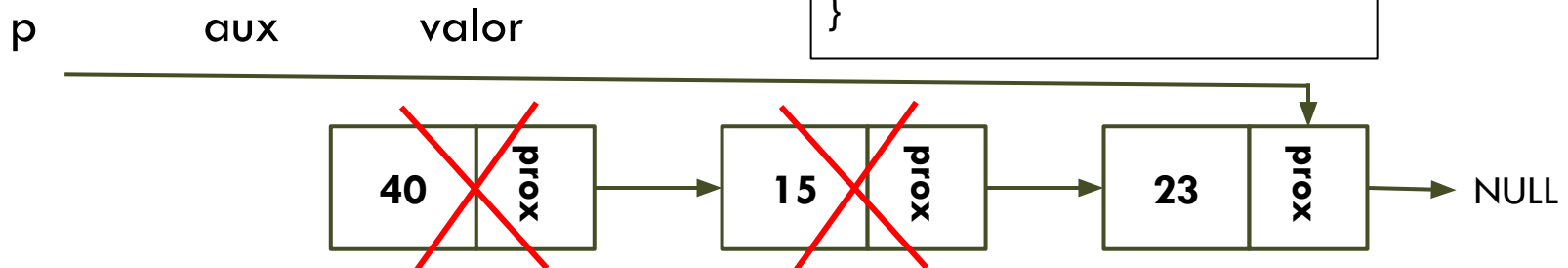
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

135

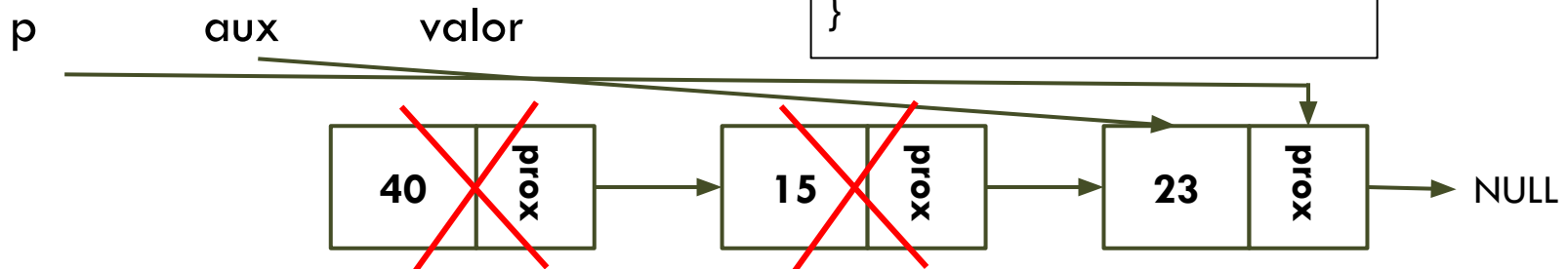
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

136

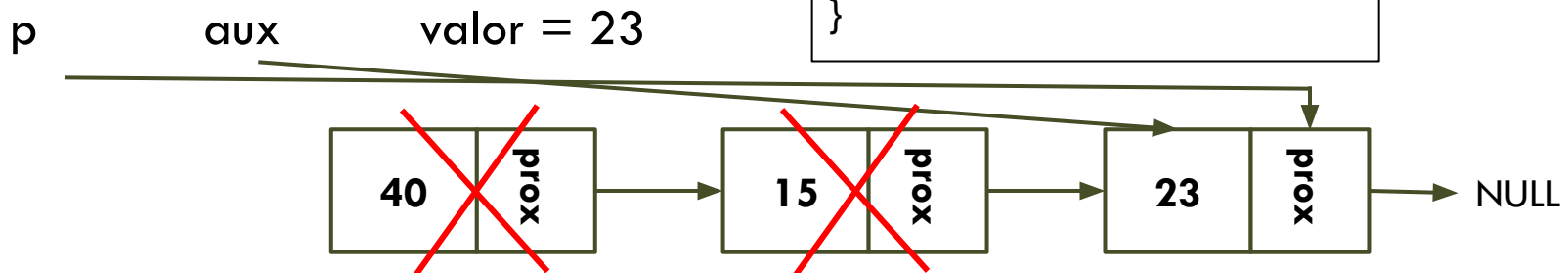
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

137

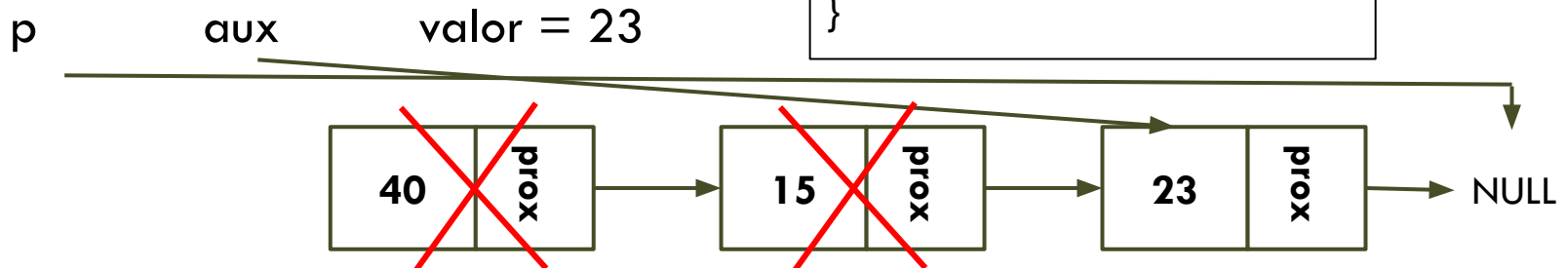
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

138

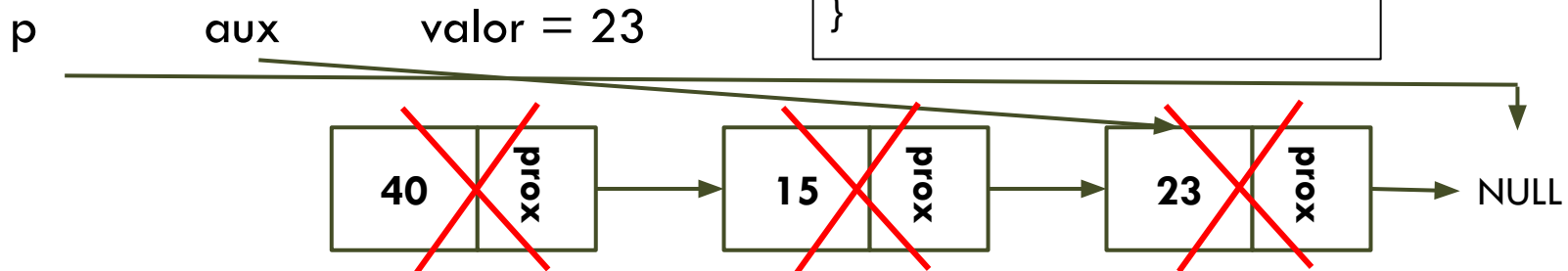
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

139

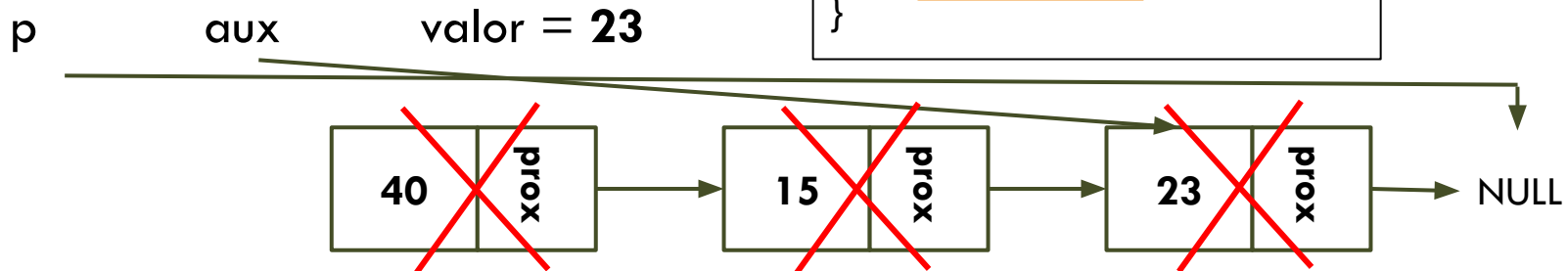
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

140

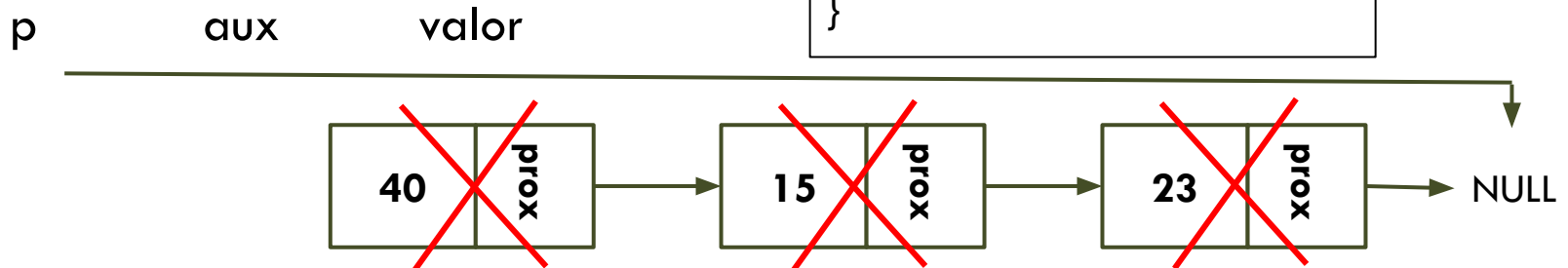
❑ Desempilhar (pop)

- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

```
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);  
pilhaD_pop(&p);
```



Implementando uma pilha dinâmica

141

❑ Desempilhar (pop)

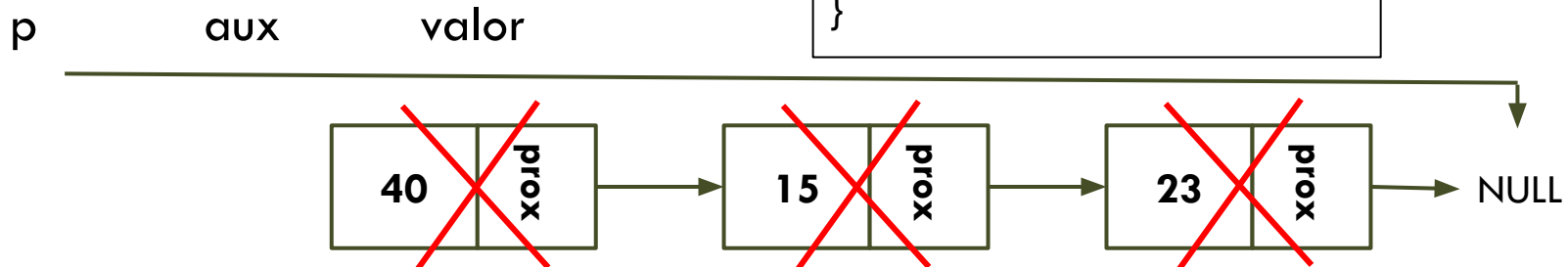
- Retira o elemento do início da lista

```
int pilhaD_vazia(PilhaD* p){  
    if(p == NULL)  
        return 1;  
    return 0;  
}
```

```
int pilhaD_pop(PilhaD** p){  
    PilhaD* aux;  
    int valor;  
    if (!pilhaD_vazia(*p)){  
        aux = *p;  
        valor = (*p)->info;  
        *p = (*p)->prox;  
        free(aux);  
    }else{  
        exit(1);  
    }  
    return valor;  
}
```

pilhaD_pop(&p);
pilhaD_pop(&p);
pilhaD_pop(&p);
pilhaD_pop(&p);

Pilha vazia!



Implementando uma pilha dinâmica

Atividade

142

❑ **Imprimir?**

- Imprimir os elementos da pilha sem usar o `pilha_pop`, pois o objetivo é só observar o conteúdo pilha sem modificar o seu conteúdo

❑ **Libera?**

- Libera a pilha depois de liberar todos os elementos (nó) alocados

Referências

143



SCHILDT, Herbert. **C completo e total**. Makron, 3ª edição revista e atualizada, 1997.



SZWARCHFITER, J. **Estruturas de Dados e seus algoritmos**. 3 ed. Rio de Janeiro: LTC, 2010.