



Algoritmos Computacionais e Estruturas de Dados

Prof. Me. Felipe Borges

Prof. Felipe Borges

Doutorando em Sistemas de Potência – UFMA – Brasil

Mestre em Sistemas de Potência – UFMA – Brasil

MBA em Qualidade e Produtividade – FAENE – Brasil

Graduado em Engenharia Elétrica – IFMA – Brasil

Graduado em Engenharia Elétrica – Fontys – Holanda

Técnico em Eletrotécnica – IFMA – Brasil

Projetos e Instalações Elétricas – Engenharia – Banco do Brasil

Desenvolvimento e Gestão de Projetos – Frencken Engineering BV

Pilhas

Um dos conceitos mais úteis na ciência da computação é o de pilha.

Nesta aula, examinaremos essa simples estrutura de dados e verificaremos por que ela desempenha esse proeminente papel nas áreas de programação e de linguagens de programação.



© Can Stock Photo – csp6591766

Pilhas



Quais operações você pode fazer com pilhas ?



Quais operações você pode fazer com pilhas ?

Empilhar



Quais operações você pode fazer com pilhas ?

Empilhar
No topo



Quais operações você pode fazer com pilhas ?

Empilhar

No topo

Desempilhar



Quais operações você pode fazer com pilhas ?

Empilhar

No topo

Desempilhar

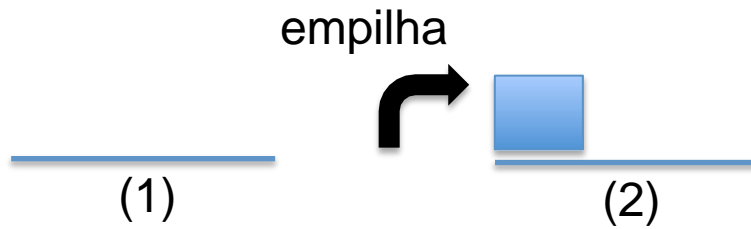
Do topo



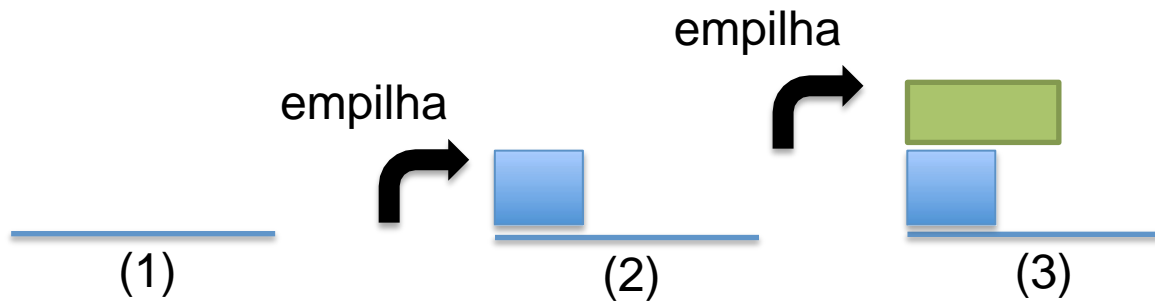
Pilha: Conceito básico

(1)

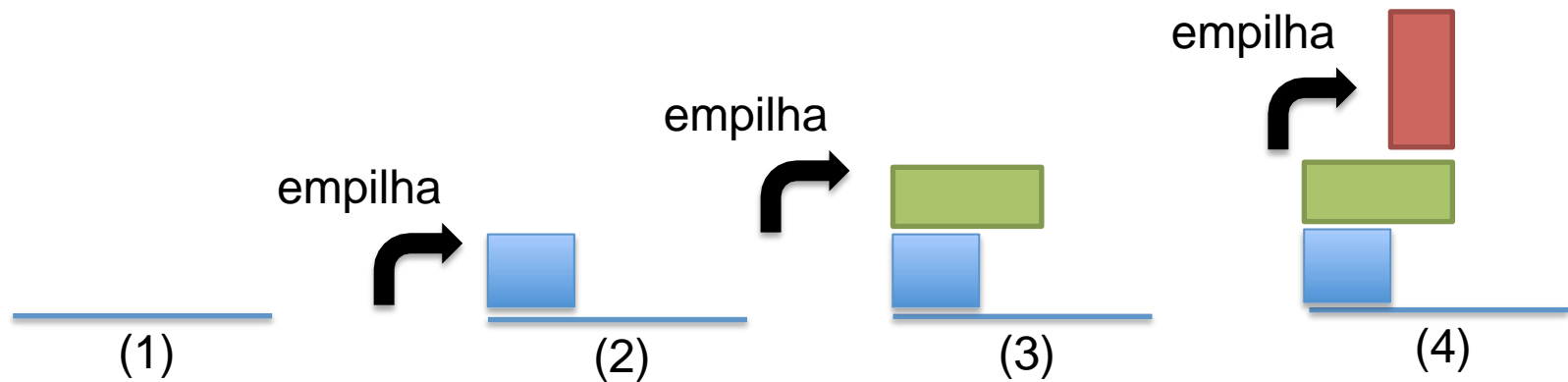
Pilha: Conceito básico



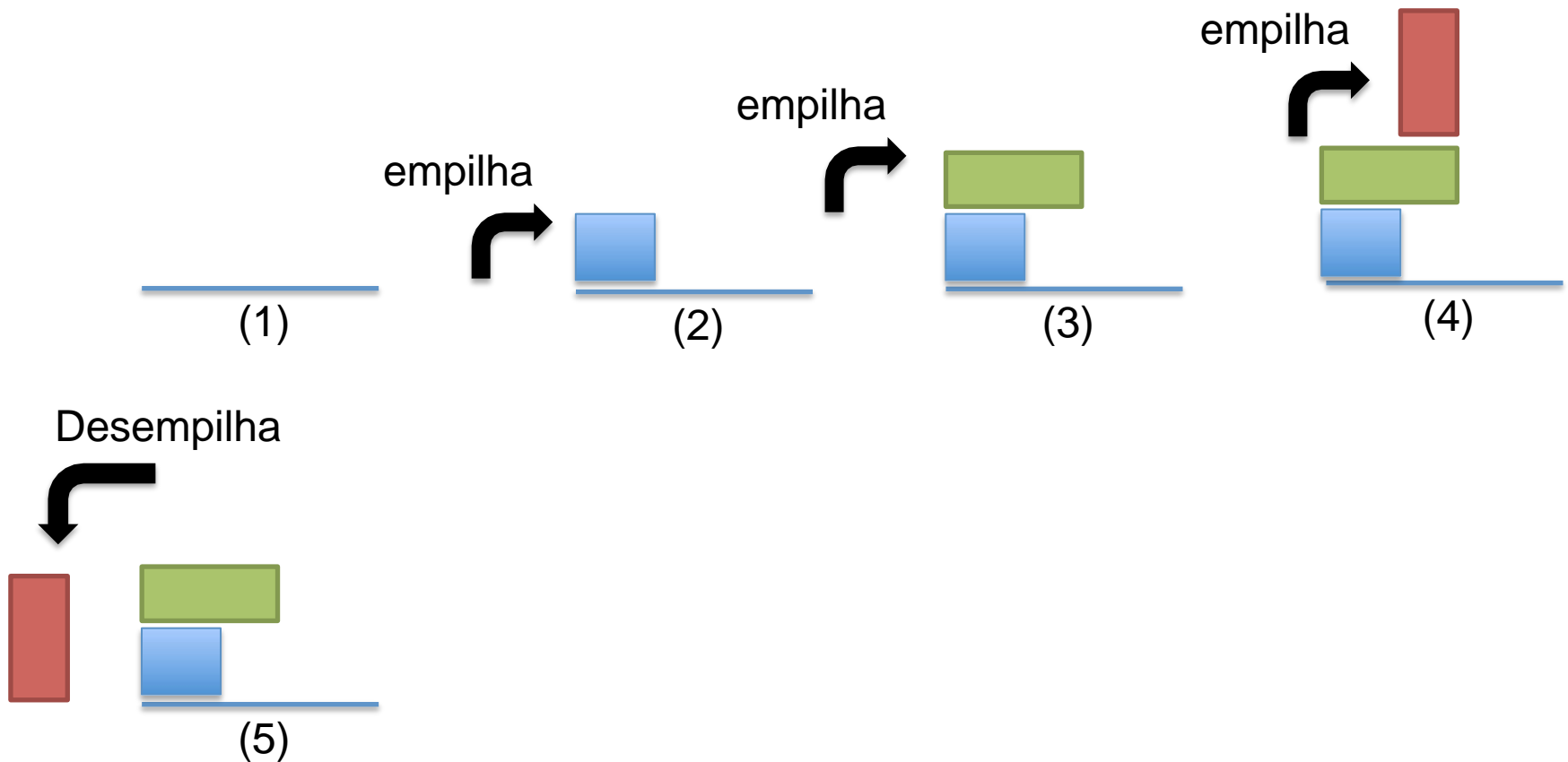
Pilha: Conceito básico



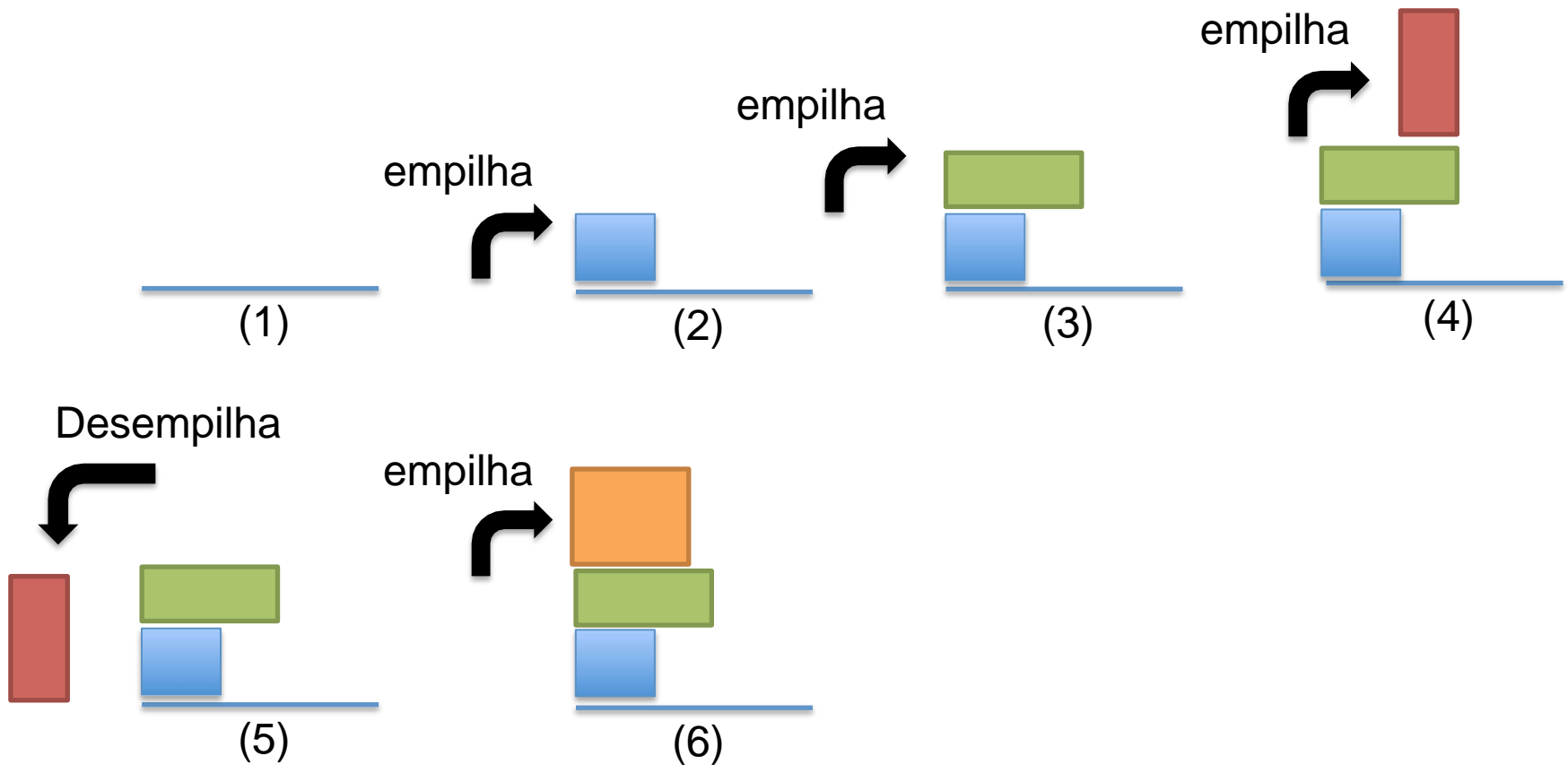
Pilha: Conceito básico



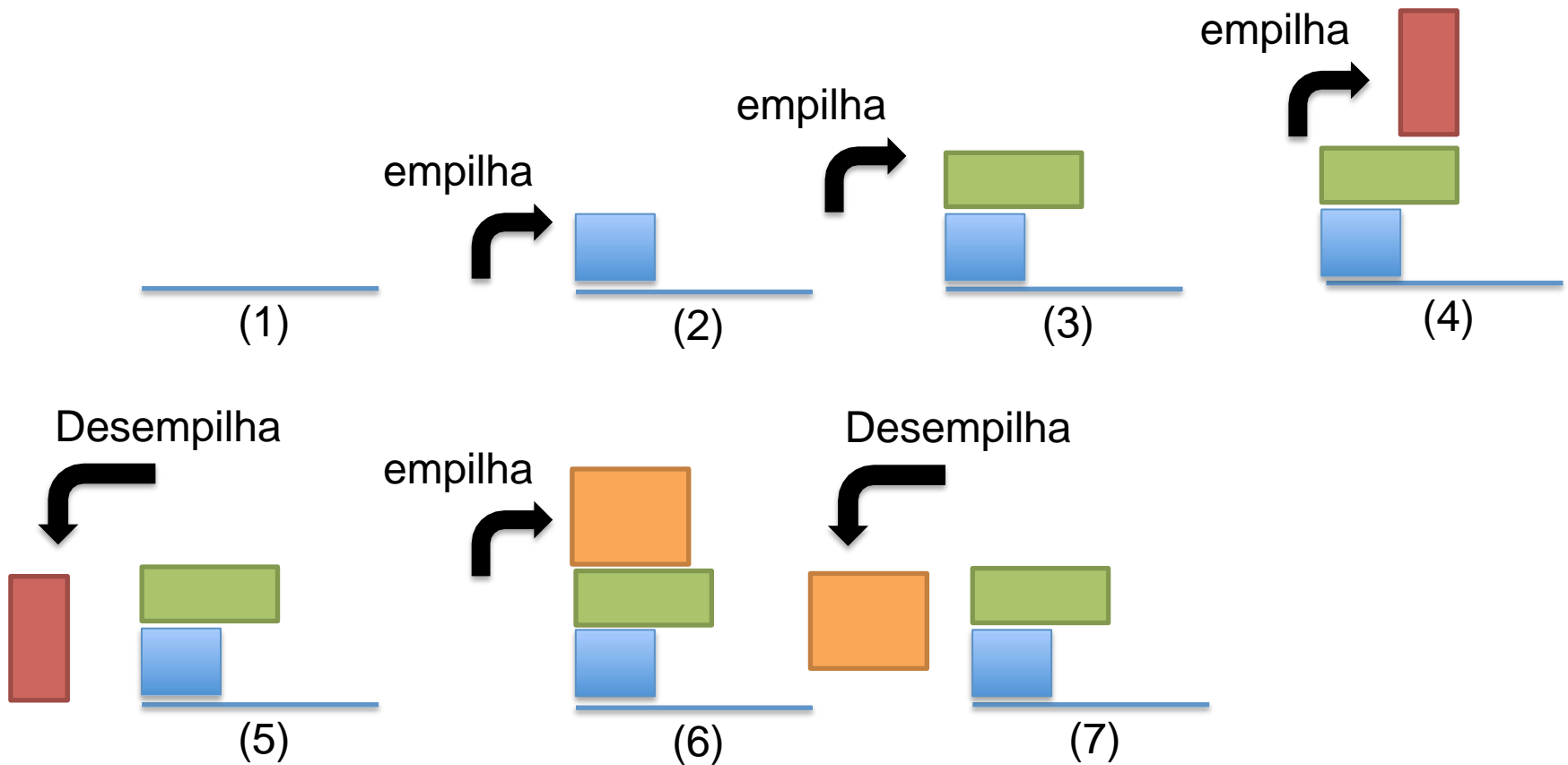
Pilha: Conceito básico



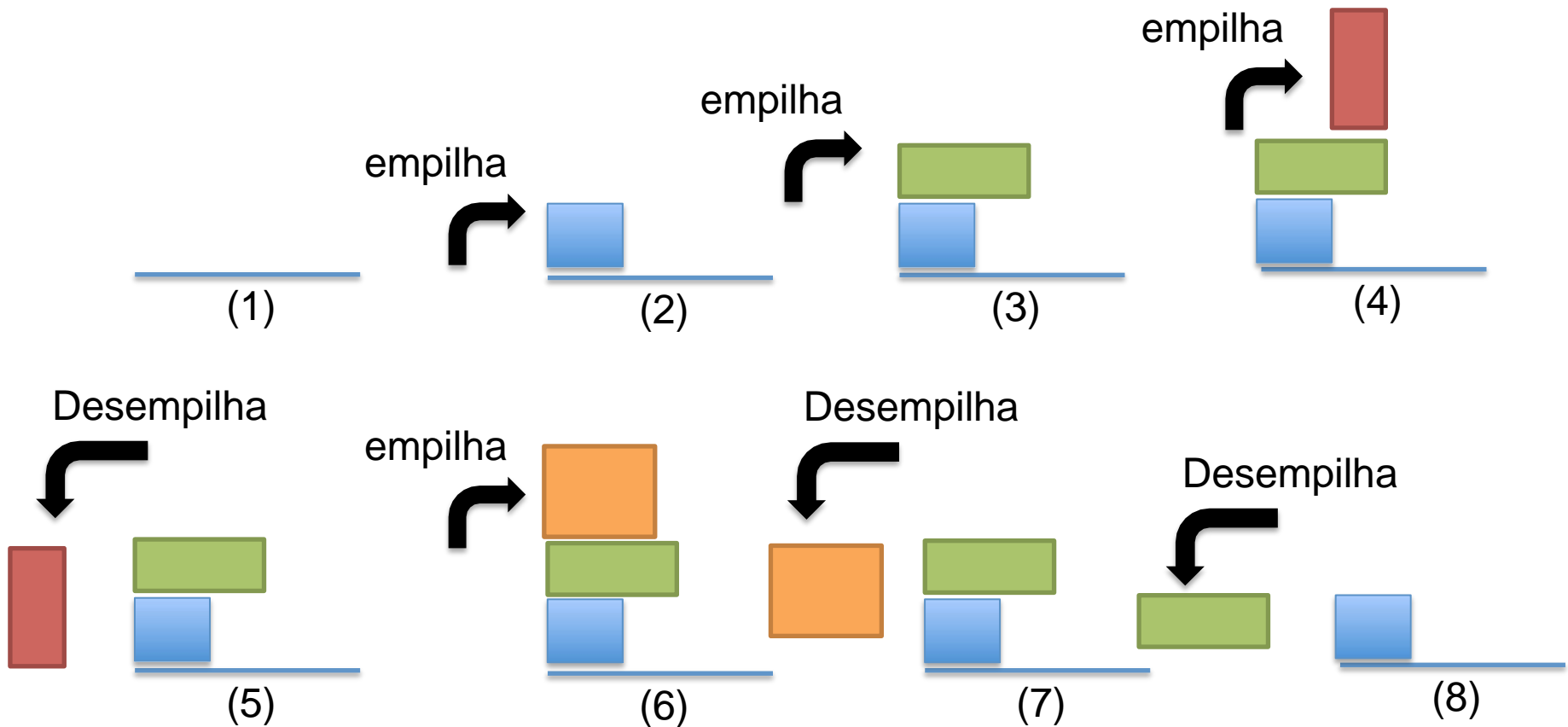
Pilha: Conceito básico



Pilha: Conceito básico

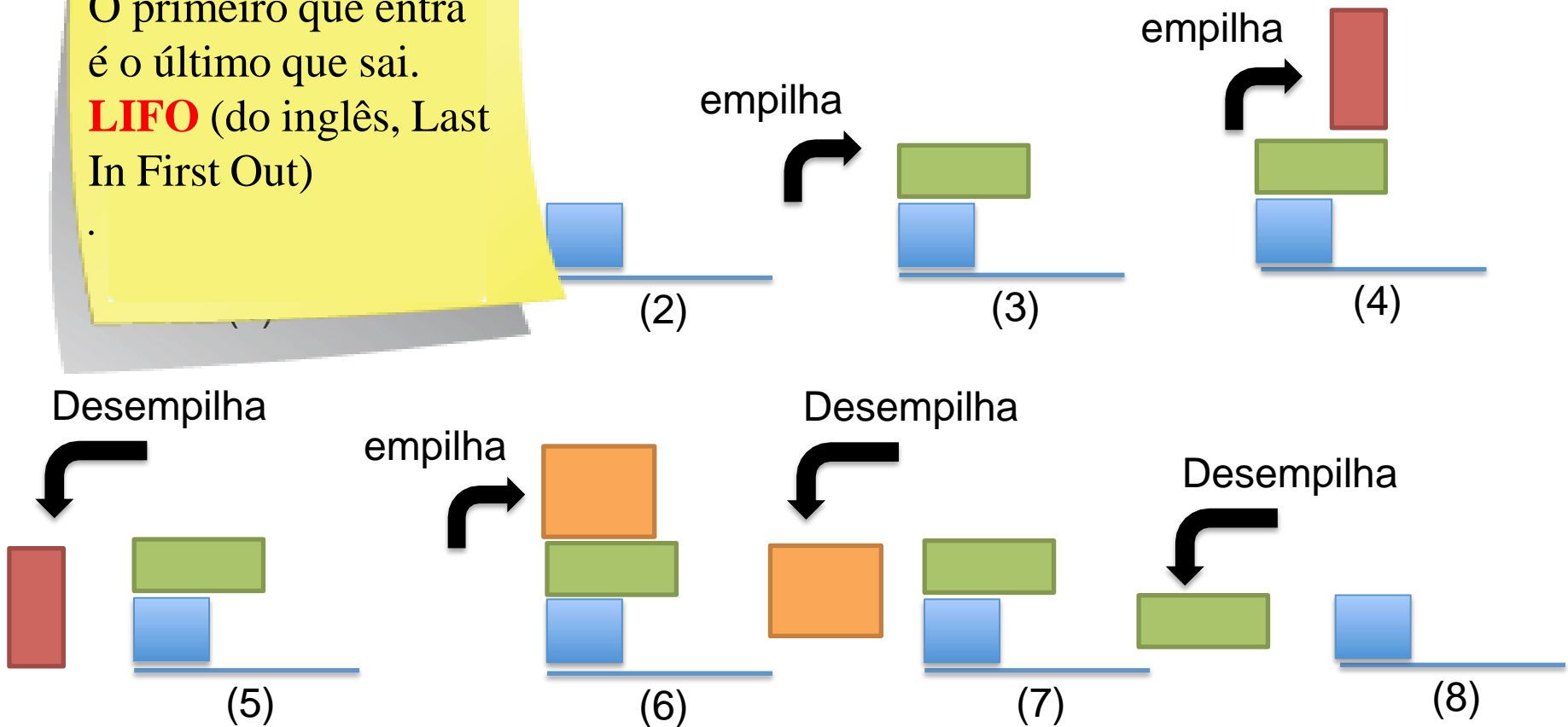


Pilha: Conceito básico



Empilha: Conceito básico

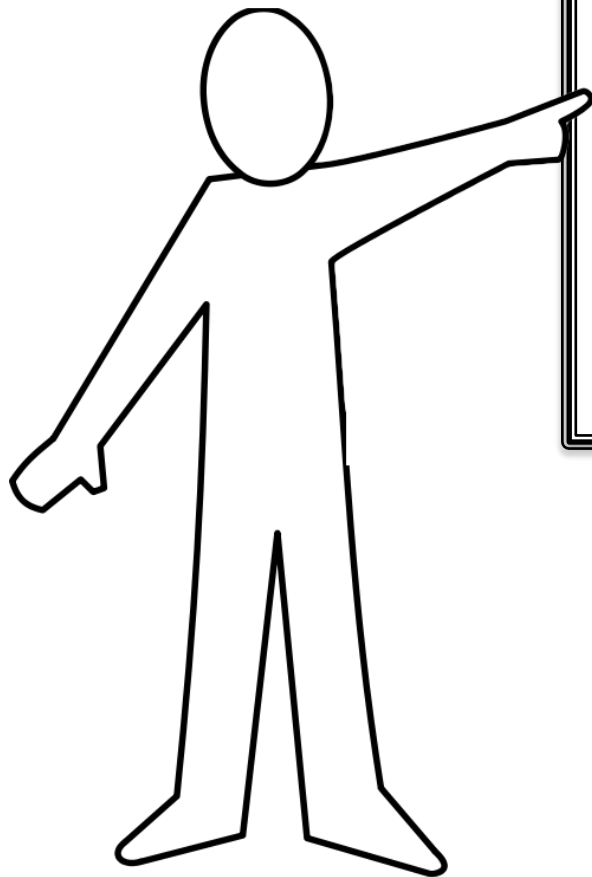
O primeiro que entra
é o último que sai.
LIFO (do inglês, Last
In First Out)



O Tipo Abstrato Pilha

O Tipo Abstrato Pilha

Ou seja, as operações que descrevem o comportamento da Pilha.



Quais seriam estas
operações, vocês são
capazes de enumerá-las ?

1. Empilhar

1. Empilhar

2. Desempilhar

1. Empilhar

2. Desempilhar

3. Verificar o valor do topo

1. Empilhar

2. Desempilhar

3. Verificar o valor do topo

4. Verificar se a pilha
está vazia

1. Empilhar

2. Desempilhar

3. Verificar o valor do topo

4. Verificar se a pilha está vazia

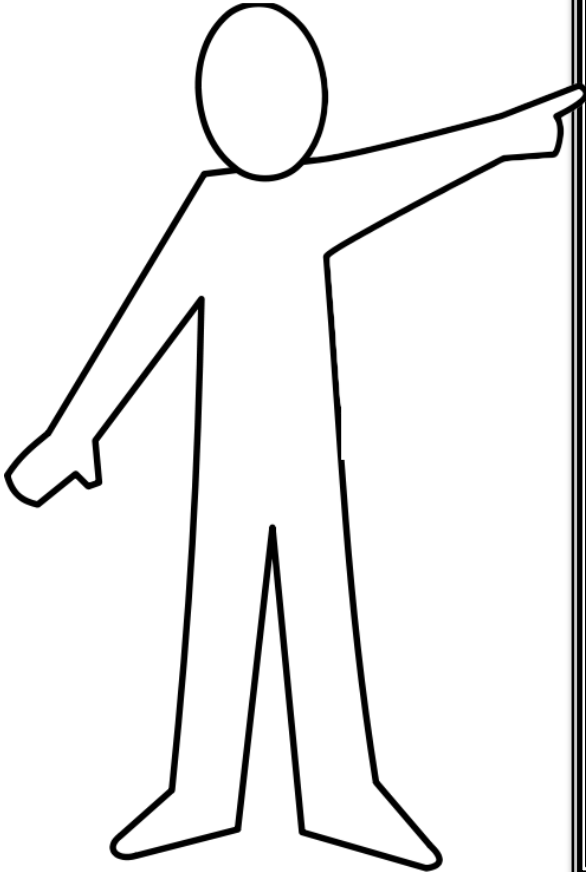
5. Cria uma Pilha

TAD: Pilha

Significado das operações (P sendo uma pilha e x um elemento)

1. *cria_pilha* () - Inicializa uma pilha, retornando uma dada pilha P
2. *empilha*(P,x) - Acrescenta o elemento x no topo da pilha P (o tamanho de P aumenta)
3. *desempilha* (P) - Remove e retorna o elemento no topo da pilha (o tamanho de P diminui)
4. *pilha_vazia* (P) - Verifica se a pilha está vazia (ou seja, não possui nenhum elemento)
5. *topo* (P) – Retorna o valor que está no topo da pilha

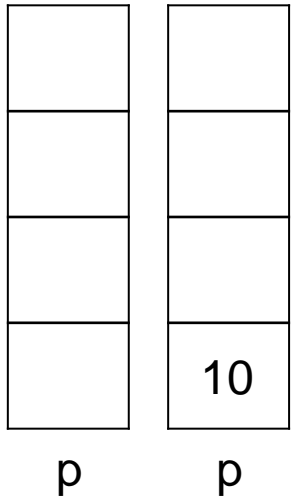
Qual será o estado final da pilha após estas operações ?



```
p = cria_pilha()  
empilha (p, 10)  
empilha (p, 20)  
empilha (p, 30)  
desempilha (p)  
empilha (p, 50)  
desempilha (p)  
desempilha (p)  
empilha (p, 90)
```

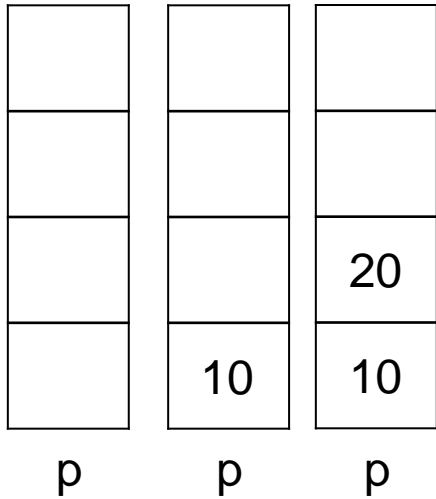


```
→ p = cria_pilha ()  
    empilha (p, 10)  
    empilha (p, 20)  
    empilha (p, 30)  
    desempilha (p)  
    empilha (p, 50)  
    desempilha (p)  
    desempilha (p)  
    empilha (p, 90)
```

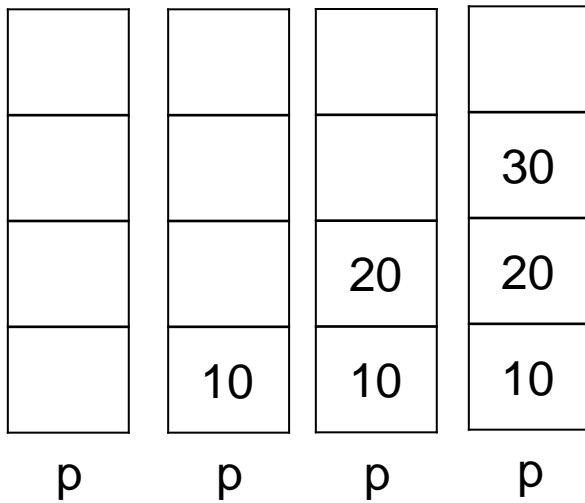



→

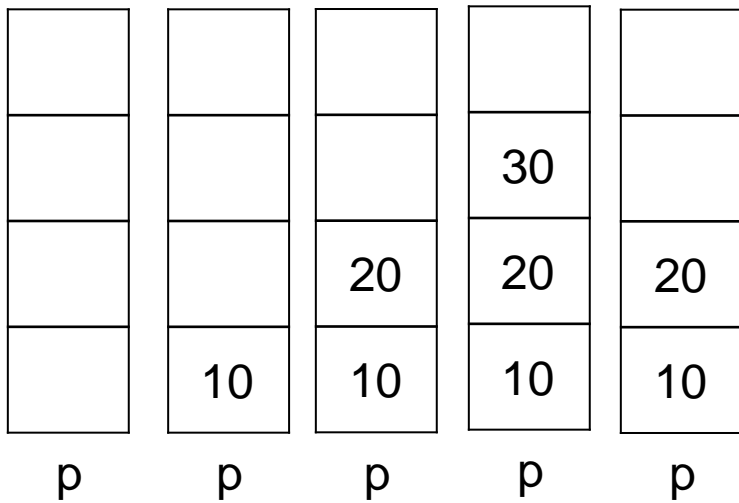
```
p = cria_pilha ()  
empilha (p, 10)  
empilha (p, 20)  
empilha (p, 30)  
desempilha (p)  
empilha (p, 50)  
desempilha (p)  
desempilha (p)  
empilha (p, 90)
```



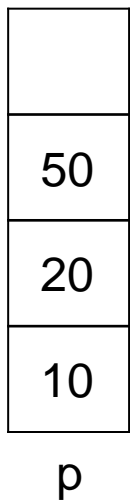
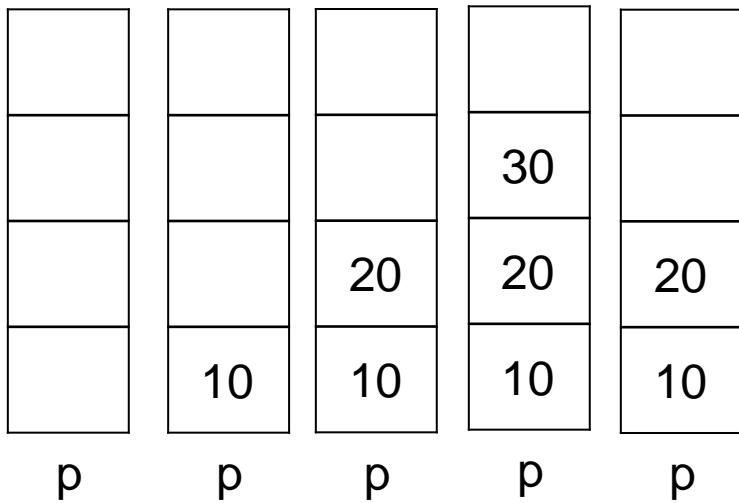
```
p = cria_pilha ()  
empilha (p, 10)  
empilha (p, 20)  
empilha (p, 30)  
desempilha (p)  
empilha (p, 50)  
desempilha (p)  
desempilha (p)  
empilha (p, 90)
```



```
p = cria_pilha ()  
empilha (p, 10)  
empilha (p, 20)  
empilha (p, 30)  
desempilha (p)  
empilha (p, 50)  
desempilha (p)  
desempilha (p)  
empilha (p, 90)
```



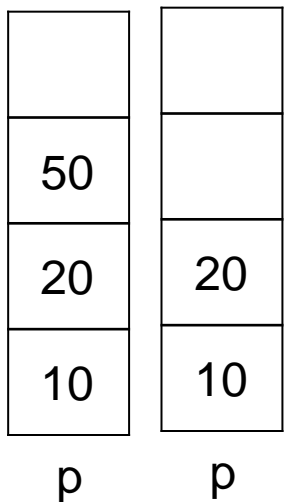
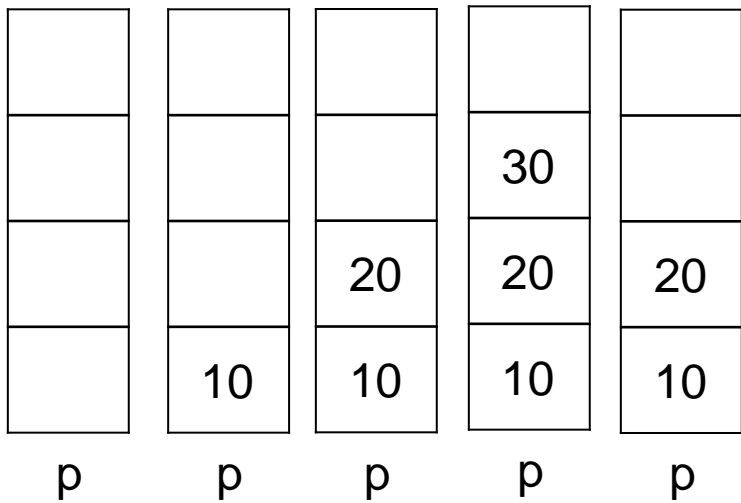
```
p = cria_pilha ()  
empilha (p, 10)  
empilha (p, 20)  
empilha (p, 30)  
desempilha (p)  
empilha (p, 50)  
desempilha (p)  
desempilha (p)  
empilha (p, 90)
```



```

p = cria_pilha ()
empilha (p, 10)
empilha (p, 20)
empilha (p, 30)
desempilha (p)
→ empilha (p, 50)
desempilha (p)
desempilha (p)
empilha (p, 90)

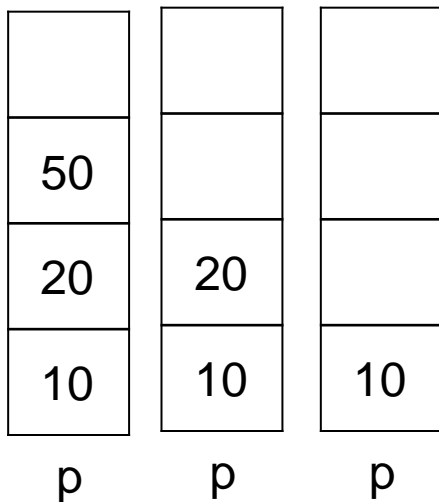
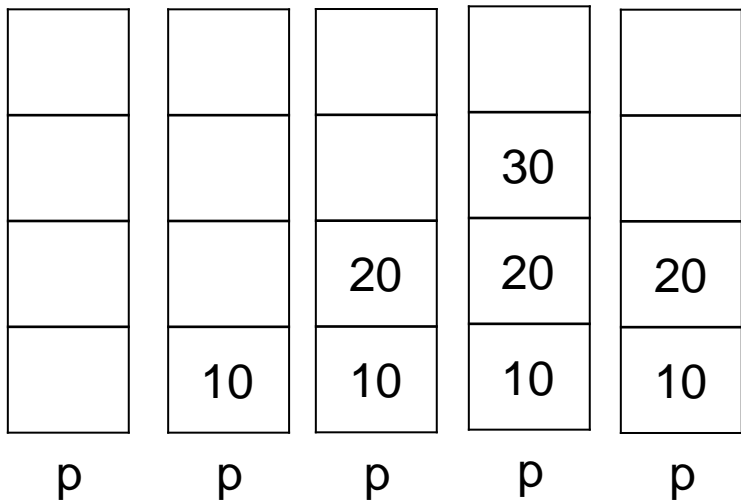
```



```

p = cria_pilha ()
empilha (p, 10)
empilha (p, 20)
empilha (p, 30)
desempilha (p)
empilha (p, 50)
→ desempilha (p)
desempilha (p)
empilha (p, 90)

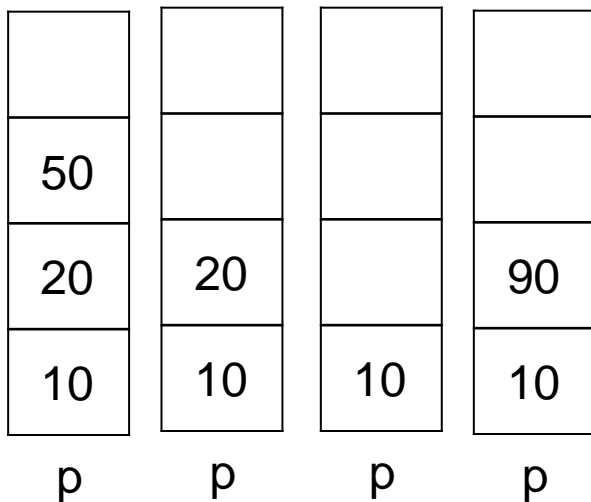
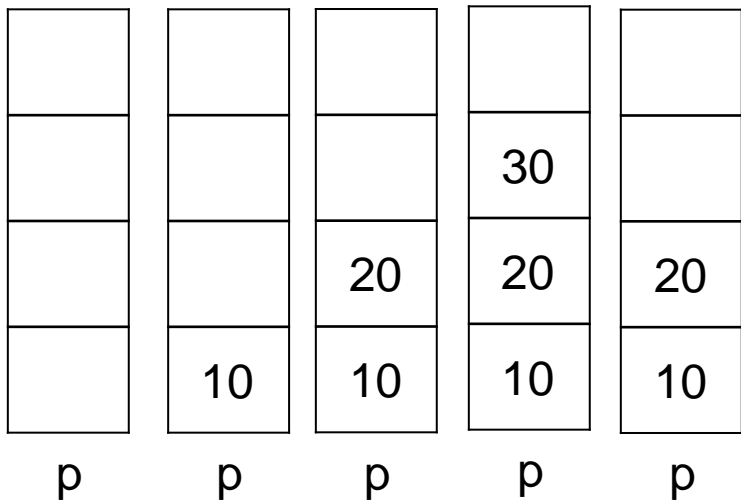
```



```

p = cria_pilha ()
empilha (p, 10)
empilha (p, 20)
empilha (p, 30)
desempilha (p)
empilha (p, 50)
desempilha (p)
desempilha (p)
empilha (p, 90)

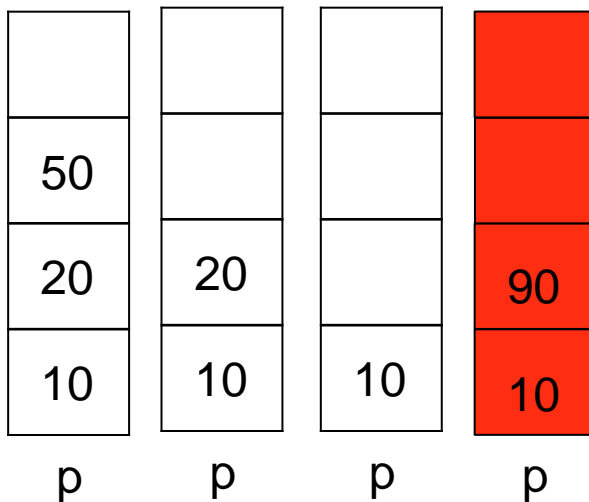
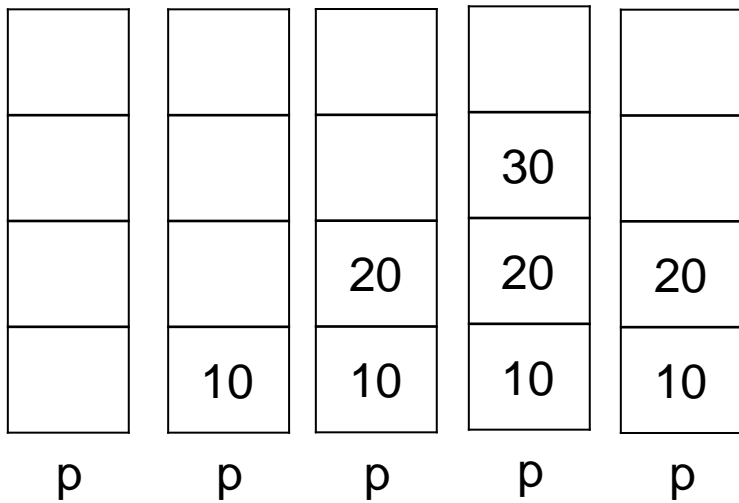
```

```

p = cria_pilha ()
empilha (p, 10)
empilha (p, 20)
empilha (p, 30)
desempilha (p)
empilha (p, 50)
desempilha (p)
desempilha (p)
→ empilha (p, 90)

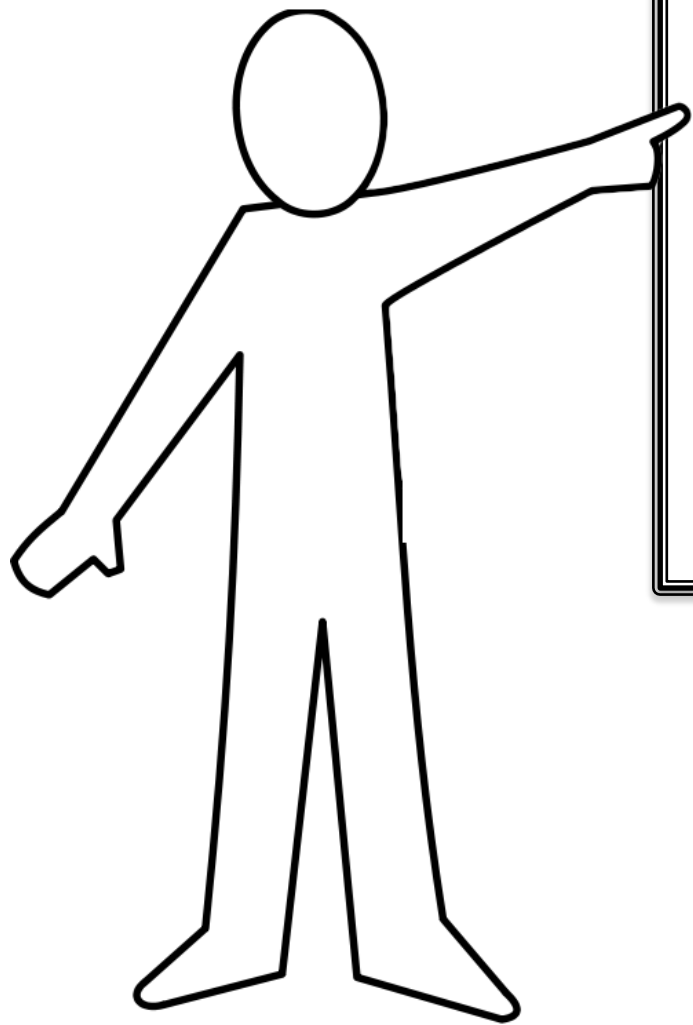
```



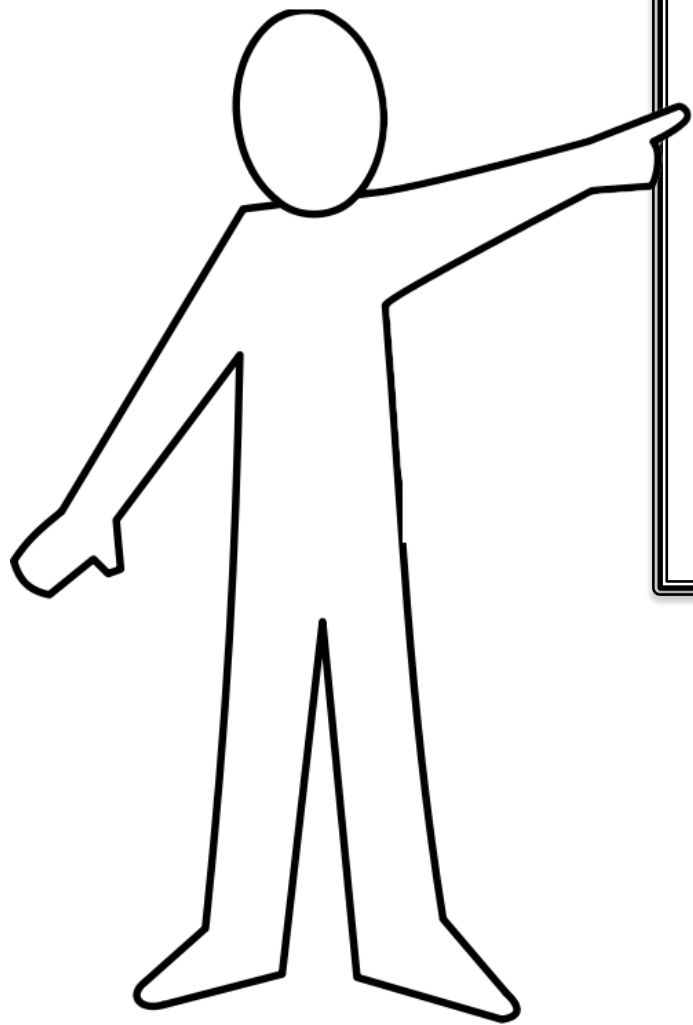
```

p = cria_pilha ()
empilha (p, 10)
empilha (p, 20)
empilha (p, 30)
desempilha (p)
empilha (p, 50)
desempilha (p)
desempilha (p)
→ empilha (p, 90)

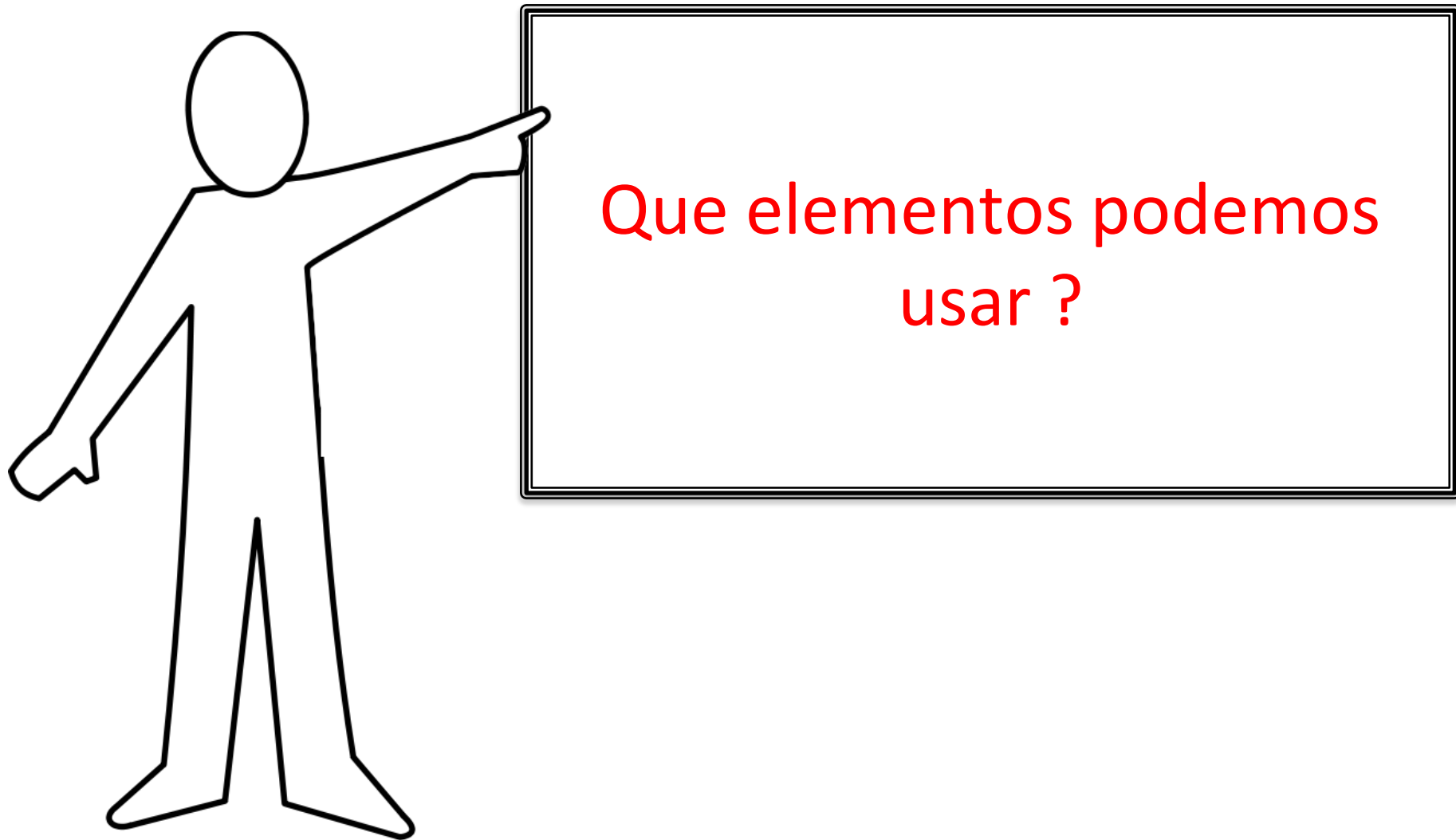
```



Entenderam o conceito da
pilha ?



Como podemos codificar
esta estrutura ?



TAD: Pilha

- Uma possibilidade é usar vetores, dados que estes permitem armazenar uma coleção de dados.

TAD: Pilha

- Uma possibilidade é usar vetores, dados que estes permitem armazenar uma coleção de dados.
- Uma pilha que usa vetor como estrutura básica é chamada de pilha estática.

TAD: Pilha

- Uma possibilidade é usar vetores, dados que estes permitem armazenar uma coleção de dados.
- Uma pilha que usa vetor como estrutura básica é chamada de pilha estática.
- Na Unidade II, veremos como codificar uma pilha dinâmica usando listas encadeadas.

Codificando o TAD Pilha

- Antes de prosseguir, pause a aula e tentem rascunhar em papel como seria as operações:
 - Empilha e
 - Desempilha

Codificando o TAD Pilha

- Ao usar um vetor como estrutura para a nossa pilha precisamos:
 - Saber em qual posição do vetor o elemento deve ser empilhado.
 - Saber quando a pilha encheu, ou seja, precisaremos de uma operação que verifica se a pilha encheu.

Codificando o TAD Pilha

- O que vamos precisar para codificar a nossa pilha?

Codificando o TAD Pilha

- O que vamos precisar para codificar a nossa pilha?

Registros

Codificando o TAD Pilha

- O que vamos precisar para codificar a nossa pilha?

Registros Funções

Codificando o TAD Pilha

- O que vamos precisar para codificar a nossa pilha?

Registros Funções Ponteiros

Codificando o TAD Pilha

- O que vamos precisar para codificar a nossa pilha?

Registros Funções Ponteiros

Alocação Dinâmica

Modificando o TAD Pilha

Dica: Caso necessário
revisem estes conceitos
usando o material de
Linguagem de
Programação I

recisar para codificar a nossa pilha?

Funções

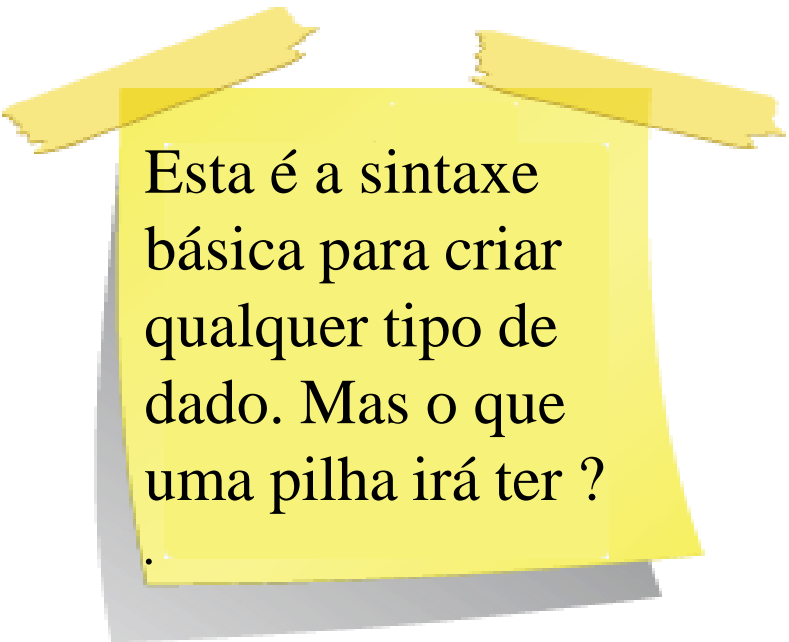
Ponteiros

Alocação Dinâmica

Codificando o TAD Pilha

- Criando o tipo de dado Pilha ...

```
typedef struct {  
  
} Pilha;
```

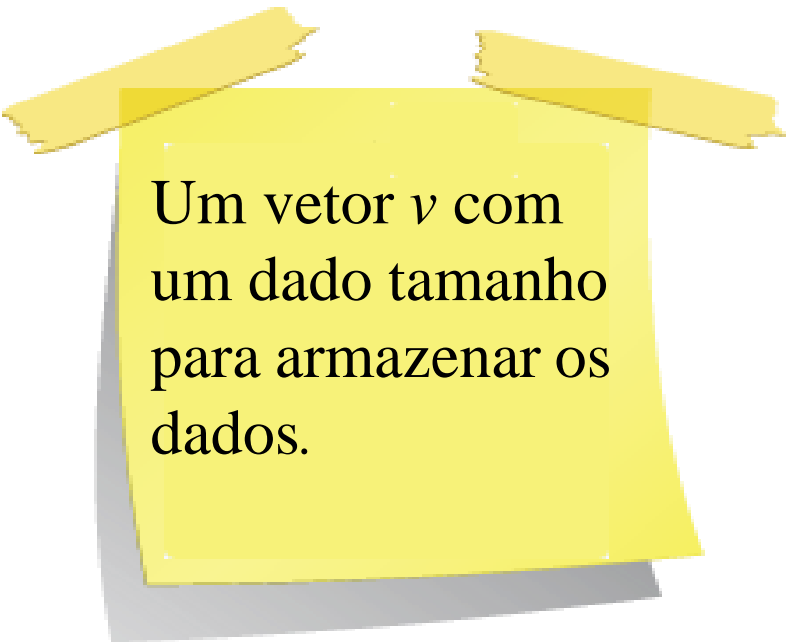


Esta é a sintaxe básica para criar qualquer tipo de dado. Mas o que uma pilha irá ter ?

Codificando o TAD Pilha

- Criando o tipo de dado Pilha ...

```
typedef struct {  
    int v[10];  
  
} Pilha;
```



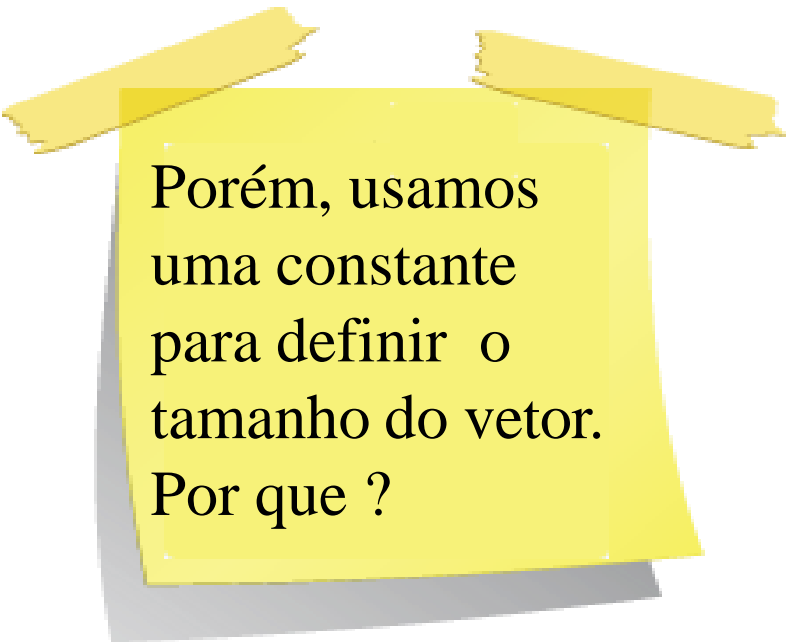
Um vetor v com um dado tamanho para armazenar os dados.

Codificando o TAD Pilha

- Criando o tipo de dado Pilha ...

```
#define MAX 10
typedef struct {
    int v[MAX];

} Pilha;
```

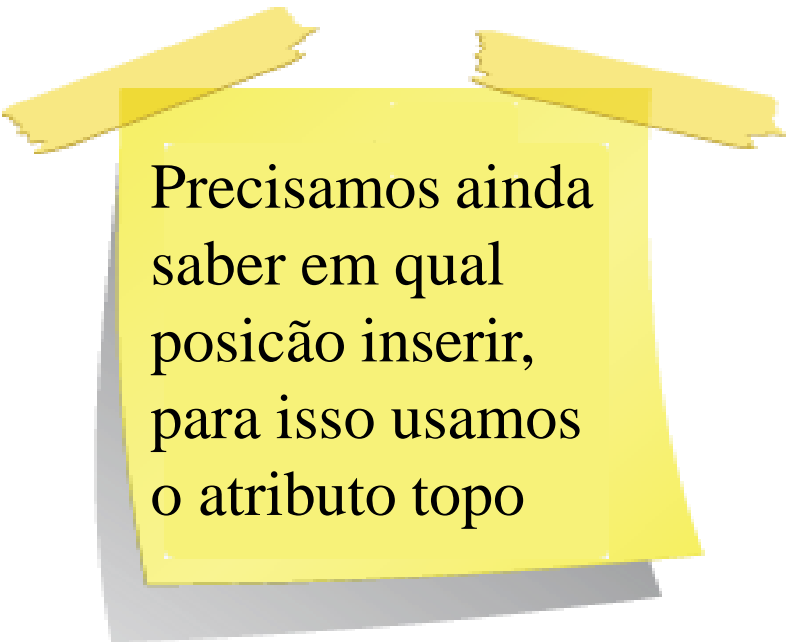


Porém, usamos
uma constante
para definir o
tamanho do vetor.
Por que ?

Codificando o TAD Pilha

- Criando o tipo de dado Pilha ...

```
#define MAX 10
typedef struct {
    int v[MAX];
    int topo;
} Pilha;
```



Precisamos ainda saber em qual posição inserir, para isso usamos o atributo topo

APLICAÇÕES

Aplicações

- Como o uso das pilhas é possível simplificar diversos algoritmos.
 - O próprio sistema operacional utiliza pilha para tratar as chamadas a funções.
 - Uma outra aplicação comum é na avaliação de expressões e de parênteses.

Identificando palíndromos

- Palíndromos são palavras ou frases que são iguais quando lidas de frente para trás
- Exemplos:
 - ANA
 - ARARA
 - ROTOR
 - SOCORRAM ME SUBI NO ONIBUS EM MARROCOS (não considerando os espaços em branco)

Identificando palíndromos

- Algoritmo

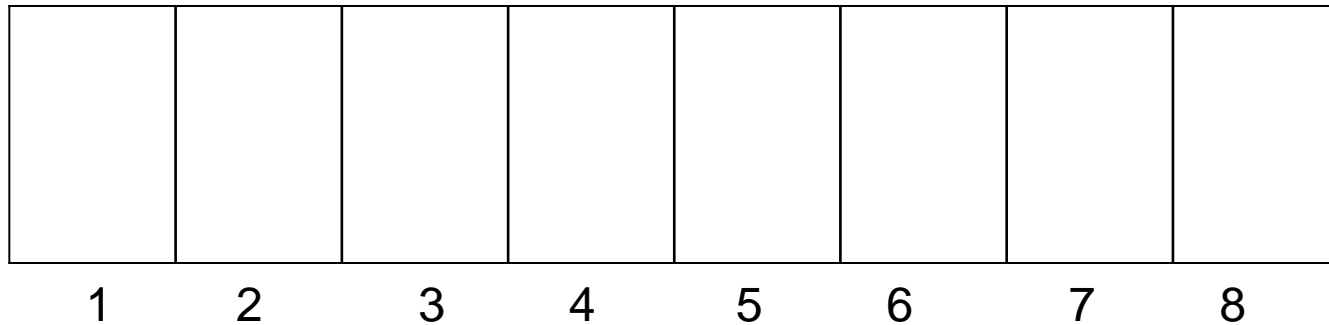
1. Empilhe $n/2$ letras
2. Descarta letra central se houver
3. Repita
 1. Compare o topo da pilha com a proxima letra
 2. Se são iguais entao Desempilhe
 3. Senao Não é um palíndromo
4. Se no final a pilha está vazia entao a palavra é um palíndromo

Identificando palíndromos

ARARA

Topo = 0

Base = 1

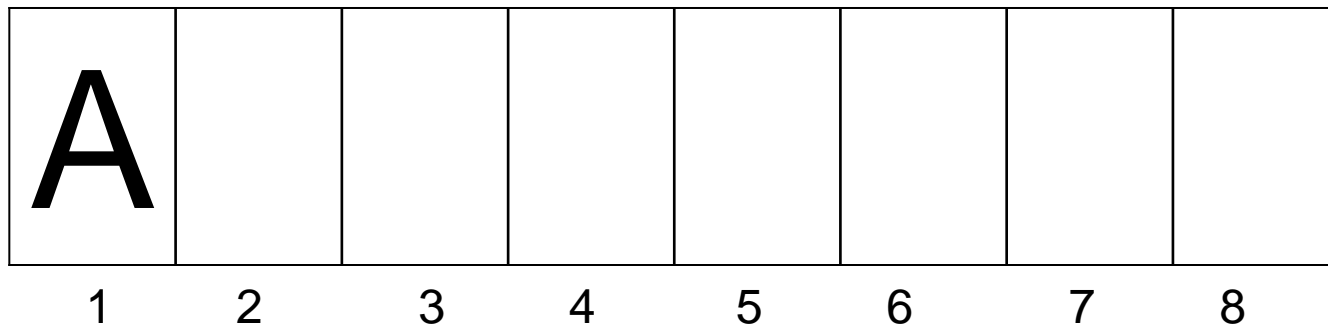


Identificando palíndromos

ARARA

Topo = 1

Base = 1



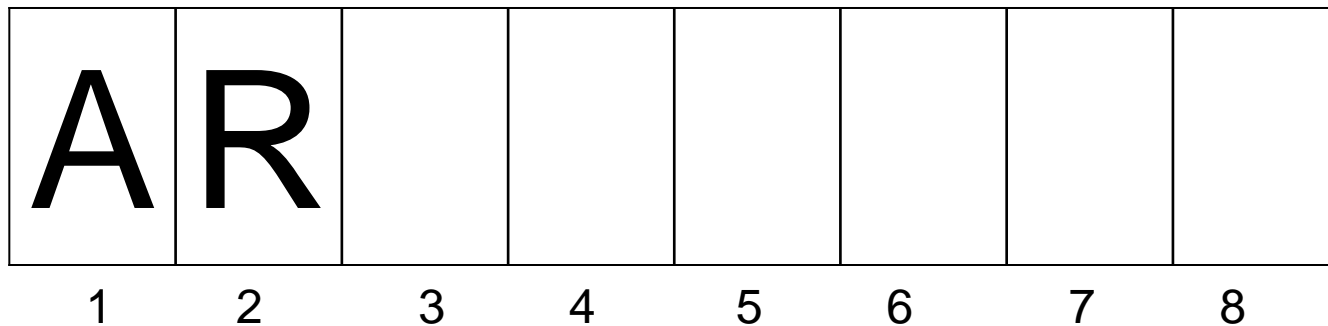
Empilha letra 1

Identificando palíndromos

ARARA

Topo =2

Base =1



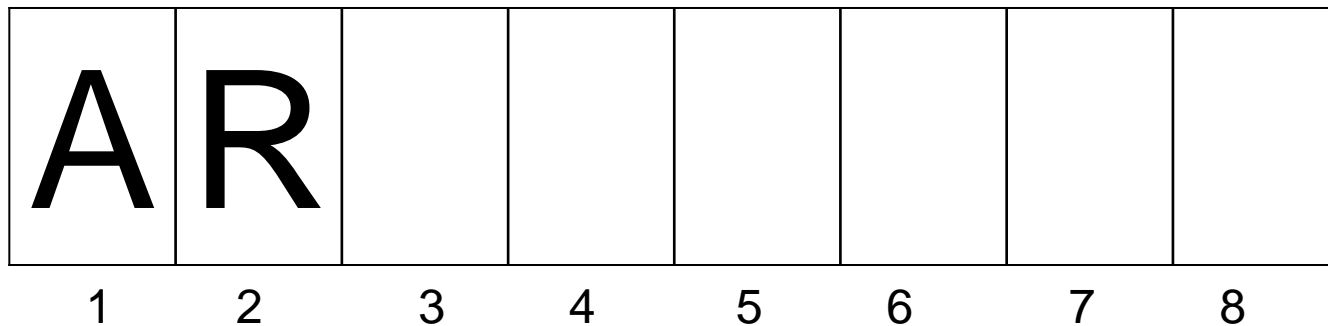
Empilha letra 2

Identificando palíndromos

ARARA

Topo = 2

Base = 1



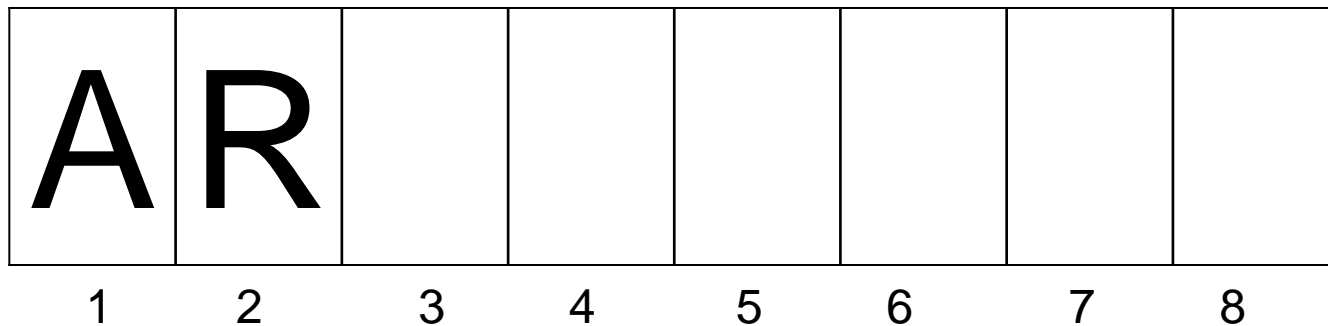
Descarta letra central

Identificando palíndromos

ARARA

Topo = 2

Base = 1



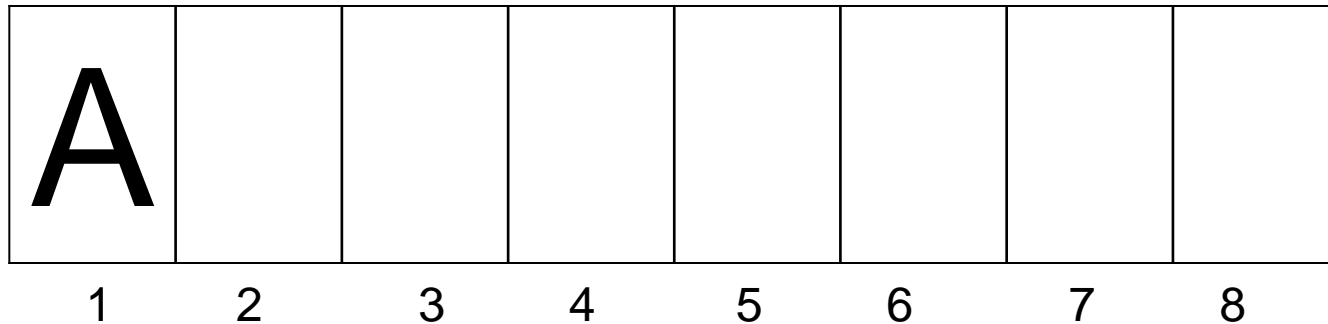
Comparar topo com próxima letra

Identificando palíndromos

ARARA

Topo = 1

Base = 1



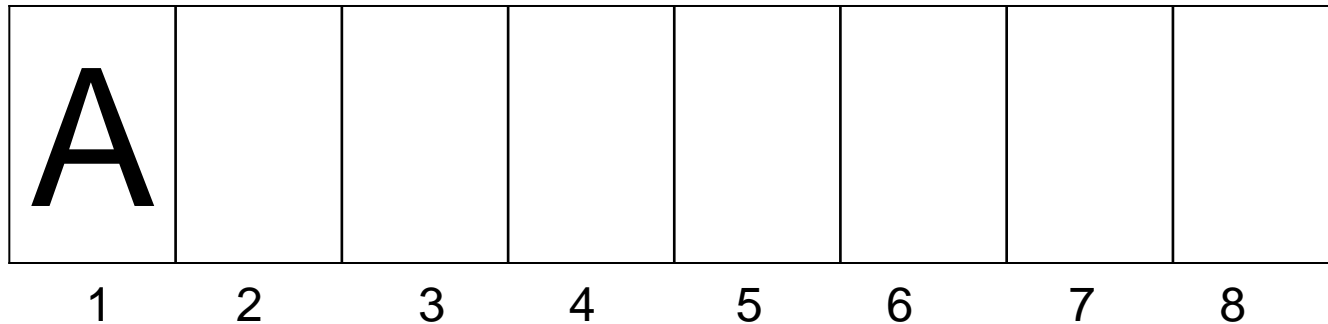
Se são iguais então desempilhe

Identificando palíndromos

ARARA

Topo = 1

Base = 1



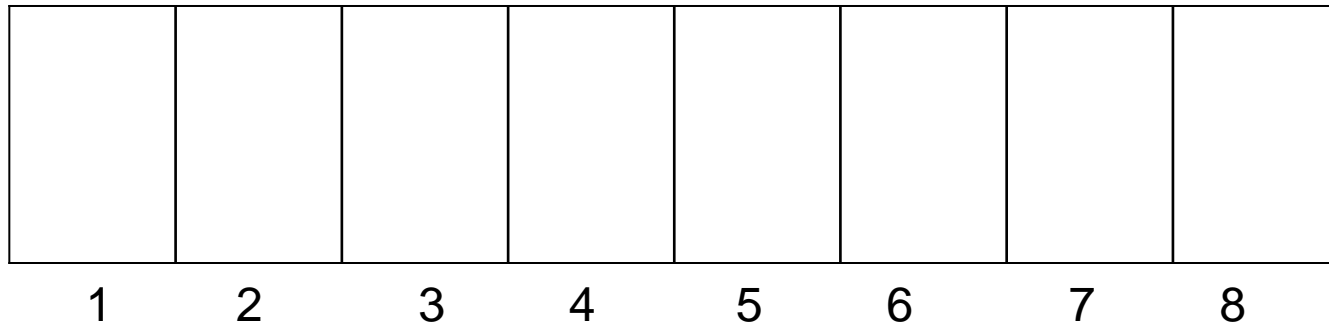
Comparar topo com próxima letra

Identificando palíndromos

ARARA

Topo = 1

Base = 1



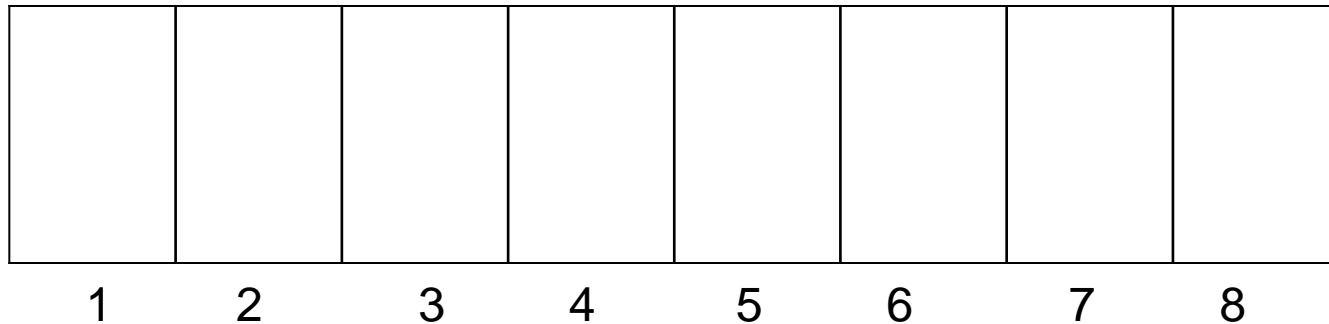
Se são iguais então desempilhe

Identificando palíndromos

ARARA

Topo = 1

Base = 1



Se no final a pilha está vazia então é um palíndromo

Outros exemplos

- Pilhas podem ser utilizadas para verificar se os parênteses em uma expressão está balanceado.

Exemplo:

$((3+4 + (4*9)$ **ERRO: Faltam dois parênteses fechando!**

Como?

Outros exemplos

- Pilhas podem ser utilizadas para verificar se os parênteses em uma expressão está balanceado.

Exemplo:

$((3+4 + (4*9)$ **ERRO: Faltam dois parênteses fechando!**

Como?

Para cada carácter da expressão faça

1. Se encontrou um "(" empilha
2. Se encontrou um ")" então
 1. Se a pilha estiver vazia: expressão invalida
 2. Caso contrario desempilhe
3. Se no final pilha estiver vazia, expressão válida.

Notação Infixa - Avaliação

Operadores entre operandos.

$$4 + 6 * 8$$

Como decidir quais operadores são avaliados primeiros ?

- Abordagem usual é usar as mesmas **regras** da matemática.
- Caso a **precedência** seja a mesma, avalia-se da esquerda para direita
- Parenteses tem **precedência** sobre todos operadores.

Notação Polonesa

É fácil observar que devido a estas regras, o algoritmo para avaliar expressões na notação infixa não é muito trivial.

O matemático polonês Jan Łukasiewicz criou uma notação em torno de 1920 que elimina a necessidade de regras.

Ficou sendo conhecida como notação polonesa.

Não é muito usado na matemática convencional, mas muito usado nas ciências da computação devido sua simplicidade.

Avaliação de expressões - Notações

Infixa: $A + (B * C) / D$

Posfixa (polonesa reversa): $A B C * + D /$

Préfixa (polonesa): $+ A / * B C D$

Avaliação de expressões - Notações

Infixa: $A + (B * C) / D$

Posfixa (polonesa reversa): $A B C * + D$

Préfixa (polonesa): $+ A / * B C D$

Exemplos de uso da notação polonesa



```
RAD XYZ HEX R= 'X'  
> Eng. Mecânico> 10 53 AUG:01  
5:  
4:  
3:  
2:  
1:  
STACK MEM BRCH TEST TYPE LIST
```

Linguagens:

Scheme e Lisp

Algoritmo de avaliação de expressão na notação pos-fixa.

Notação Posfixa

A avaliação de expressões nesta notação é a mais simples. Percorrendo uma expressão da esquerda para direita, sabemos que ele deve operar os dois últimos valores encontrados (considerando que os operadores são binários, por exemplo os aritméticos).

Notação posfixa

Considerem a seguinte expressão $7\ 3\ +\ 5\ *$.

Expressão	Elemento	Ação	Pilha
$73+5^*$			P:[]
$3+5^*$	7	Empilhar	P:[7]
$+5^*$	3	Empilhar	P:[3,7]
5^*	+	Desempilha 3 Desempilha 7 Empilha 3+7	P:[7] P:[] P:[10]
*	5	Empilhar	P:[5,10]
	*	Desempilha 5 Desempilha 10 Empilha $5*10$	P:[10] P:[] P:[50]

Se a pilha tiver mais de um elemento, erro, caso contrário retorna o único valor da pilha.

Referências Bibliográficas

- AARON, Tanenbaun. **Estruturas de Dados usando C.** São Paulo: Makron Books, 1995.
- PEREIRA, Silvio. **Estruturas de Dados Fundamentais.** São Paulo: Editora Erica, 1996.
- VELOSO, Paulo et al. **Estruturas de Dados.** Rio de Janeiro: Editora Campus, 1983.

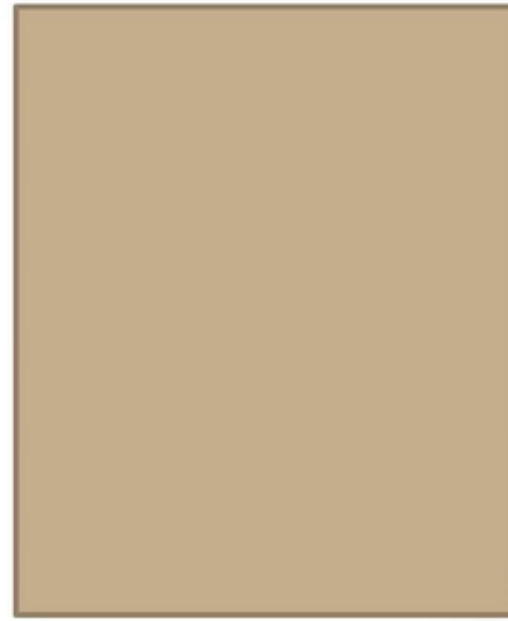
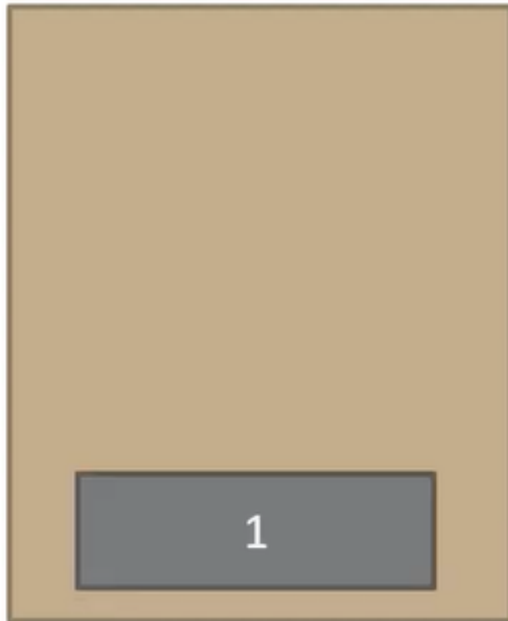
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
 - Similar a uma pilha de livros
 - LIFO – Last In, First Out
 - O último a entrar é o primeiro a sair



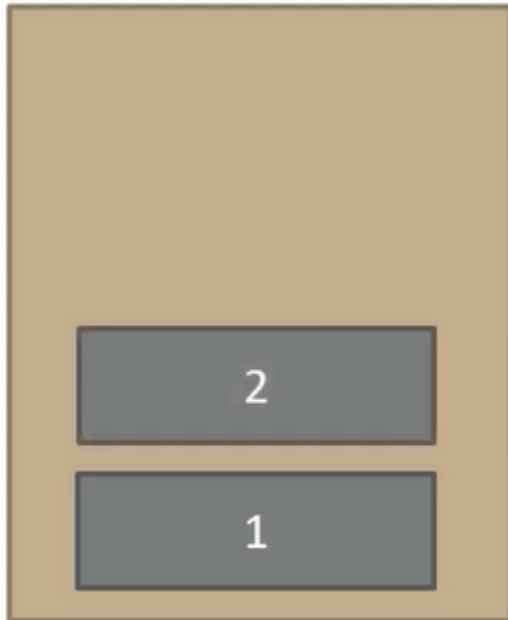
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
 - Similar a uma pilha de livros
 - LIFO – Last In, First Out
 - O último a entrar é o primeiro a sair



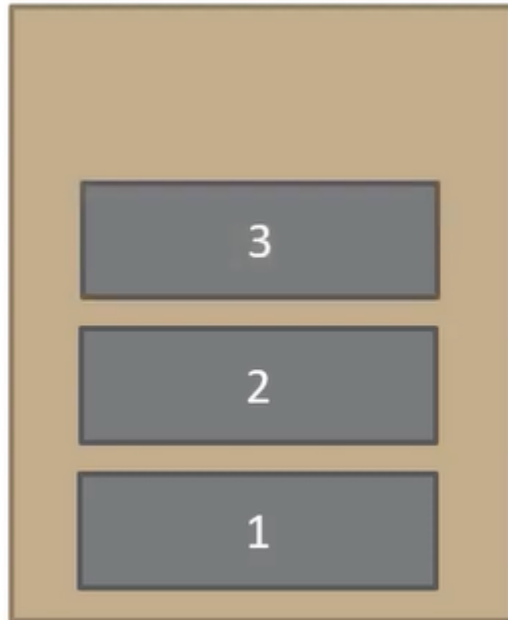
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
 - Similar a uma pilha de livros
 - LIFO – Last In, First Out
 - O último a entrar é o primeiro a sair



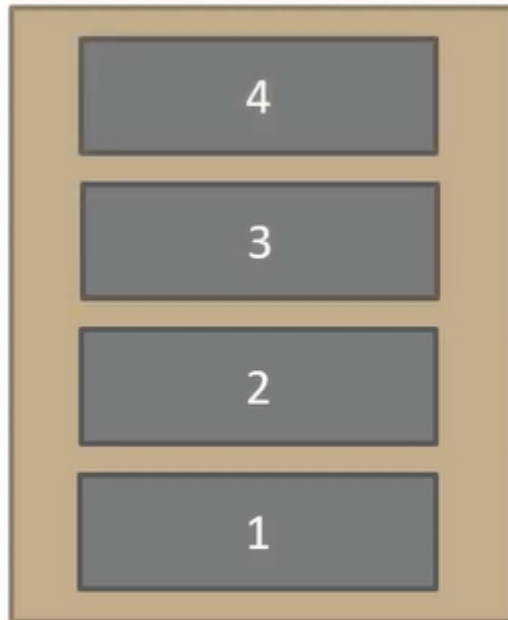
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
 - Similar a uma pilha de livros
 - LIFO – Last In, First Out
 - O último a entrar é o primeiro a sair



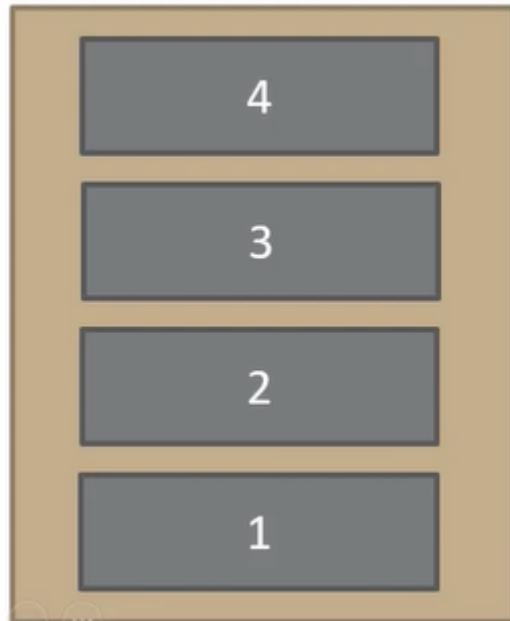
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
 - Similar a uma pilha de livros
 - LIFO – Last In, First Out
 - O último a entrar é o primeiro a sair



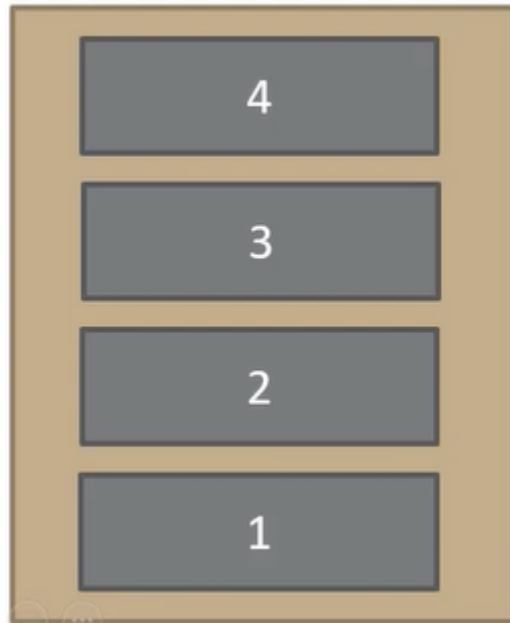
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
 - Similar a uma pilha de livros
 - LIFO – Last In, First Out
 - O último a entrar é o primeiro a sair



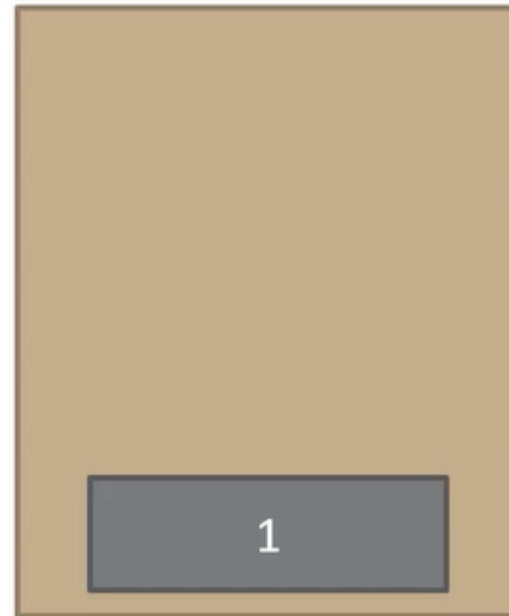
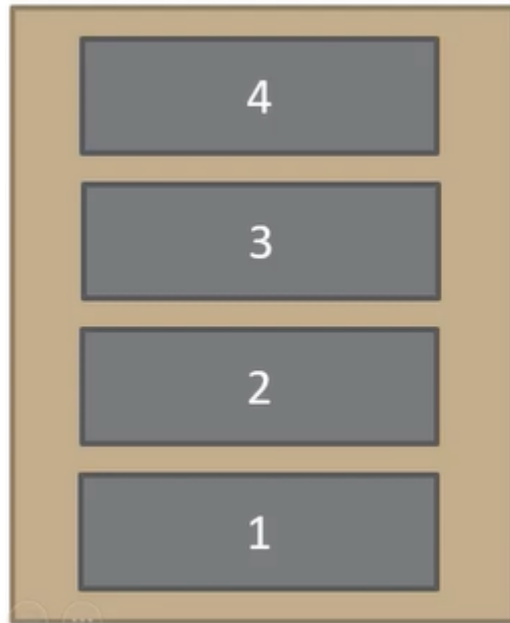
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
 - Similar a uma pilha de livros
 - LIFO – Last In, First Out
 - O último a entrar é o primeiro a sair



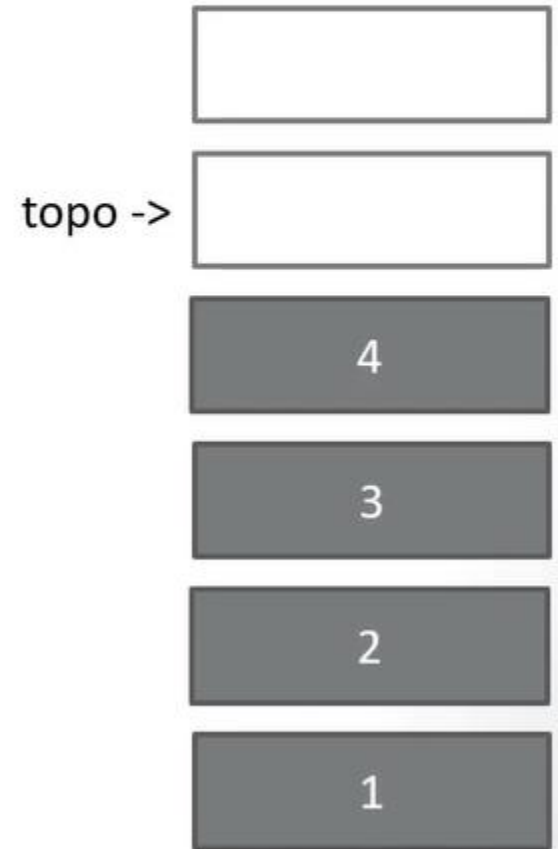
PILHA (stack)

- Representa um tipo de estrutura onde novos elementos são empilhados sobre os antigos
 - Similar a uma pilha de livros
 - LIFO – Last In, First Out
 - O último a entrar é o primeiro a sair



PILHA (stack)

- Características
 - Número de elementos - n
 - Máximo número de elementos que cabe na pilha
 - Topo – *topo*
 - Posição onde um novo elemento entrará
- Operações
 - Empilhar
 - Adiciona um novo elemento ao topo
 - Incrementa o topo
 - Stack overflow (pilha cheia) se $topo = n$
 - Desempilhar
 - Remover um elemento
 - Decrementa o topo
 - Stack empty (pilha vazia) se $topo = 0$



PILHA (stack)

- Pode ser implementada com vetor
- Variáveis de controle da pilha
 - Número de elementos - n
 - Máximo número de elementos que cabe na pilha
 - Topo – *topo*
 - Posição onde um novo elemento entrará

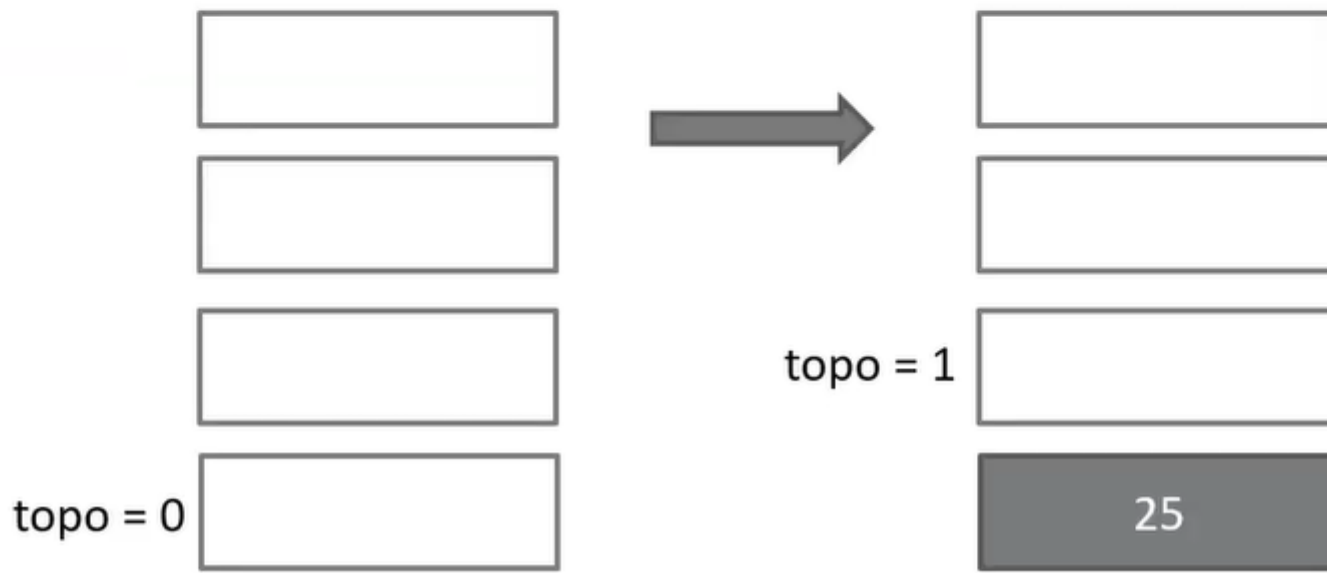
PILHA (stack)

- Aplicações
 - Controle de fluxo de programas
 - Quando o usuário chama uma função em um programa
 1. O sistema operacional adiciona a uma pilha (stack) a posição da memória de código onde o programa estava no momento da chamada da função
 2. O controle de programa salta para uma posição na memória de código onde inicia o código da função
 3. Após a função terminar (return), o sistema operacional verifica na pilha a posição da memória de código onde o programa deve continuar
 - Stack overflow
 - Ocorre quando o usuário chama muitas vezes uma função dentro de outra, a ponto da pilha encher e o programa não mais conseguir retornar
 - Pode ocorrer em microcontroladores e outros sistemas que possuem pouca memória, onde a pilha é pequena – limite de chamadas

PILHA (stack)

```
int pilha[20];           // cria uma vetor de 20 elementos  
int topo = 0;           // define onde esta o topo da pilha
```

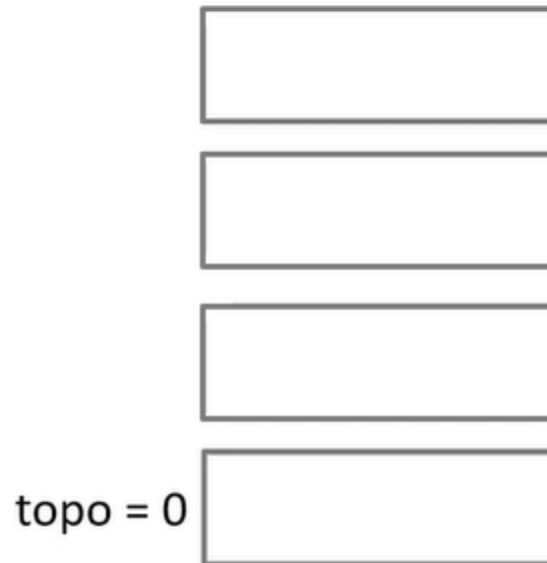
```
// empilha um elemento  
pilha[topo] = 25;  
topo++;
```



PILHA (stack)

Um exemplo de struct para uma pilha

```
typedef struct {  
    int pilha[4];    // tamanho da pilha  
    int topo;  
} Pilha;
```



Exemplo - PILHA

- Faça um programa que implemente uma pilha de 20 elementos do tipo inteiro utilizando struct
 - Implemente um método que empilhe um novo inteiro
 - O método deve enviar uma mensagem se a pilha estiver cheia
 - `void empilha(int valor, Pilha *pilha);`
 - Implemente um método que desempilhe
 - O método deve exibir uma mensagem se a pilha estiver vazia
 - `void desempilha(Pilha *pilha);`
 - Implemente um método que retorne 1 se a pilha está cheia e 0 se não
 - `int isCheia(Pilha *pilha);`
 - Implemente um método que retorne 1 se a pilha está vazia e 0 se não
 - `int isVazia(Pilha *pilha);`
- Porque nos métodos deve ser passado como parâmetro um ponteiro para a pilha, e não a pilha diretamente?

Exemplo - PILHA

- Faça um programa que implemente uma pilha de 20 elementos do tipo inteiro utilizando struct

Exemplo - PILHA

```
#include <stdio.h>
#include <stdlib.h>
#define TAMANHO_PILHA 20

typedef struct{
    int vetor[TAMANHO_PILHA]; //VETOR É A PILHA com tamanho = a tamanho da pilha
    int topo;
}Pilha;

int main(int argc, char *argv[]) {
    //DECLARA UMA PILHA
    Pilha p;
    p.topo=0;
    //empilhar numero 45: Faz-se dessa forma pq p foi declarado de forma local
    p.vetor[p.topo]=45;
    p.topo++;
    //desempilhar o ultimo numero
    p.topo--;
    printf("Elemento retirado da pilha: %d \n",p.vetor[p.topo]);
    return 0;
}
```

Exemplo - PILHA

- Faça um programa que implemente uma pilha de 20 elementos do tipo inteiro utilizando struct
 - Implemente um método que empilhe um novo inteiro
 - O método deve enviar uma mensagem se a pilha estiver cheia
 - `void empilha(int valor, Pilha *pilha);`

Exemplo - PILHA

```
#include <stdio.h>
#include <stdlib.h>
#define TAMANHO_PILHA 20
typedef struct{
    int vetor[TAMANHO_PILHA]; //tamanho da pilha
    int topo;
}Pilha;

//prototipo da função empilha
void empilha(int valor,Pilha *pilha){
    //pilha->topo significa: ponteiro "pilha" apontando para CONTEÚDO de um item de
    uma struct
    if(pilha->topo < TAMANHO_PILHA){ //verificando se a pilha não está cheia
        //daí pode empilhar
        pilha->vetor[pilha->topo]=valor;
        pilha->topo++;
    }else{
        printf("Nao ha mais espaco na pilha, \n");
    }
}
```

Exemplo - PILHA

- Porque nos métodos deve ser passado como parâmetro um ponteiro para a pilha, e não a pilha diretamente?

Devido as variáveis locais e globais.

Ao não passar como ponteiro, as funções vão manipular a cópia da pilha, mas não vão manipular a pilha original (da função *main*)

Exemplo - PILHA

- Faça um programa que implemente uma pilha de 20 elementos do tipo inteiro utilizando struct
 - Implemente um método que empilhe um novo inteiro
 - O método deve enviar uma mensagem se a pilha estiver cheia
 - `void empilha(int valor, Pilha *pilha);`
 - Implemente um método que desempilhe
 - O método deve exibir uma mensagem se a pilha estiver vazia
 - `void desempilha(Pilha *pilha);`

Exemplo - PILHA

```
void desempilha(Pilha *pilha){
    if(pilha->topo > 0){
        pilha->topo--; //desempilha
        printf("Elemento retirado: %d. \n",pilha->vetor[pilha->topo]);
    }else{
        printf("A pilha está vazia. \n"); //pilha vazia
    }
}
```

Exemplo - PILHA

- Faça um programa que implemente uma pilha de 20 elementos do tipo inteiro utilizando struct
 - Implemente um método que empilhe um novo inteiro
 - O método deve enviar uma mensagem se a pilha estiver cheia
 - `void empilha(int valor, Pilha *pilha);`
 - Implemente um método que desempilhe
 - O método deve exibir uma mensagem se a pilha estiver vazia
 - `void desempilha(Pilha *pilha);`
 - Implemente um método que retorne 1 se a pilha está cheia e 0 se não
 - `int isCheia(Pilha *pilha);`
 - Implemente um método que retorne 1 se a pilha está vazia e 0 se não
 - `int isVazia(Pilha *pilha);`

Exemplo - PILHA

```
int isCheia(Pilha *pilha){  
    if(pilha->topo >= TAMANHO_PILHA){  
        return 1;           //identifica se a pilha está cheia  
    }else{  
        return 0;  
    }  
}
```

Exemplo - PILHA

```
int isVazia(Pilha *pilha){  
    if(pilha->topo == 0){  
        return 1;           //identifica se a pilha está vazia  
    }else{  
        return 0;  
    }  
}
```

Exemplo - PILHA

INSERINDO ITENS NA PILHA E VERIFICANDO A POSIÇÃO QUE ESTÁ NO TOPO.

```
int main(int argc, char *argv[]) {  
    //DECLARA UMA PILHA  
    Pilha p;  
    p.topo=0; // o topo da pilha deve começar em zero  
  
    empilha(12,&p);    //elemento 1  
    empilha(2,&p);     //elemento 2  
    empilha(6,&p);     //elemento 3  
    empilha(9,&p);     //elemento 4  
    empilha(50,&p);    //elemento 5  
  
    printf("Topo da pilha: %d. \n",p.topo);  
    return 0;  
}
```

Exemplo - PILHA

IMPLEMENTAÇÃO FUNÇÃO IMPRIMIR PILHA:

```
void imprimePilha(Pilha *pilha){  
    int i;  
    for(i=(pilha->topo);i-->0){ //valor inicial de i é a ultima posição da pilha e daí  
decrementa  
        printf("%02d \n",pilha->vetor[i]);  
    }  
}
```

Exemplo - PILHA

TESTANDO A FUNÇÃO `imprimePilha`:

```
int main(int argc, char *argv[]) {  
    //DECLARA UMA PILHA  
    Pilha p;  
    p.topo=0; // o topo da pilha deve começar em zero  
  
    empilha(12,&p);    //elemento 1  
    empilha(2,&p);     //elemento 2  
    empilha(6,&p);     //elemento 3  
    empilha(9,&p);     //elemento 4  
    empilha(50,&p);    //elemento 5  
  
    printf("Topo da pilha: %d. \n",p.topo);  
  
    imprimePilha(&p);  
  
    return 0;  
}
```

Exemplo - PILHA

TESTANDO A FUNÇÃO desempilha:

```
int main(int argc, char *argv[]) {  
    //DECLARA UMA PILHA  
    Pilha p;  
    p.topo=0; // o topo da pilha deve começar em zero  
  
    empilha(12,&p);    //elemento 1  
    empilha(2,&p);     //elemento 2  
    empilha(6,&p);     //elemento 3  
    empilha(9,&p);     //elemento 4  
    empilha(50,&p);    //elemento 5  
  
    desempilha(&p);  
  
    printf("Topo da pilha: %d. \n",p.topo);  
  
    imprimePilha(&p);  
    return 0;  
}
```


Exercício - PILHA

- Com base no exercício 01, adicione um menu para que o usuário possa escolher se quer adicionar um elemento à pilha, ou remover um elemento
 - Se pressionar 'a', o programa solicita um número a ser adicionado à pilha
 - Se pressionar 'r', o programa remove um elemento da pilha
 - Se pressionar 's', o programa exibe os elementos já presentes na pilha
 - Se pressionar 't', o programa exibe o número de elementos
 - Se pressionar 'x', encerra o programa

```
Escolha uma opção:  
a: adiciona elemento  
r: remove elemento  
s: exibe elementos  
...
```

Menu em C

```
char c = 'a';

while(c!='x')      // se o usuario pressionar 'x', encerra o laço do menu
{
    printf("Digite um caractere:\n");
    scanf("%c",&c); // captura caractere pressionado

    switch(c)
    {
        case 'a':
            ...
            break;

        case 'x':
            ...
            break;

        default:
            ...
            break;
    }
}
```

Exemplo - PILHA

```
int main(int argc, char *argv[]) {  
  
    //  
    char c = 'a';    // guarda o caractere (opcao) do usuario  
  
    // declara uma pilha  
    Pilha p;  
    p.topo = 0;  
  
    // menu infinito  
    while(c!='x')    // se o usuario pressionar 'x', encerra o laço do menu  
    {  
        printf("Digite um caractere:\n");  
        //c = (char)getchar(); // captura caractere pressionado  
        scanf("%c",&c);  
  
        switch(c)  
        {  
            case 'a':  
                printf("Empilhar\n");  
                // TODO  
                break;  
  
            case 'x':  
                printf("Sair\n");  
                // TODO  
                break;  
  
            case 's':  
                // TODO  
                break;  
  
            default:  
                // TODO  
                break;  
        }  
    }  
}
```