INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ESTRUTURA DE DADOS APLICADA

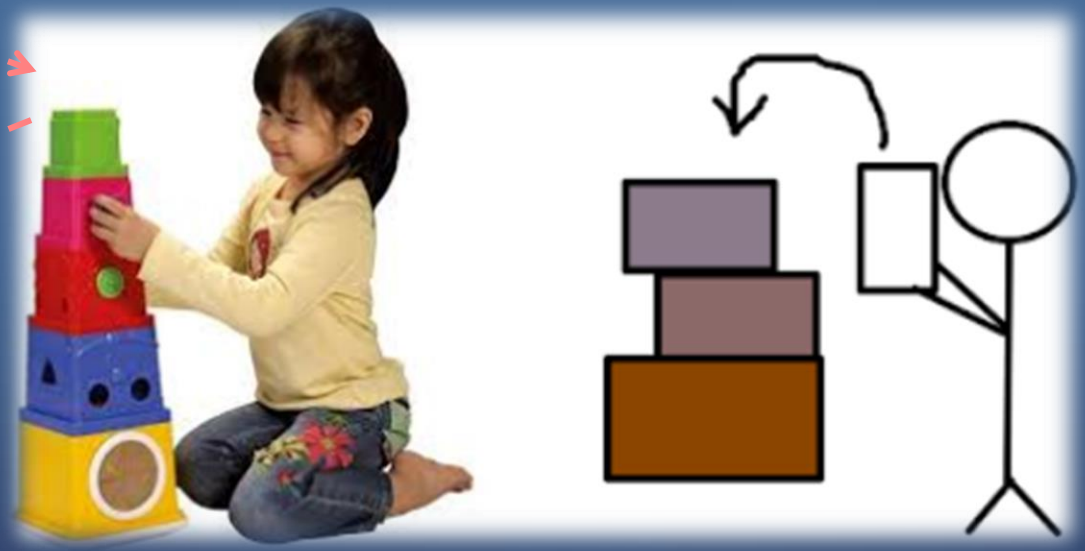
PILHA

PROFESSORA RESPONSÁVEL:
José Arnaldo Mascagni de Holanda
CONTATOS: arnaldomh@ifsp.edu.br

PILHA

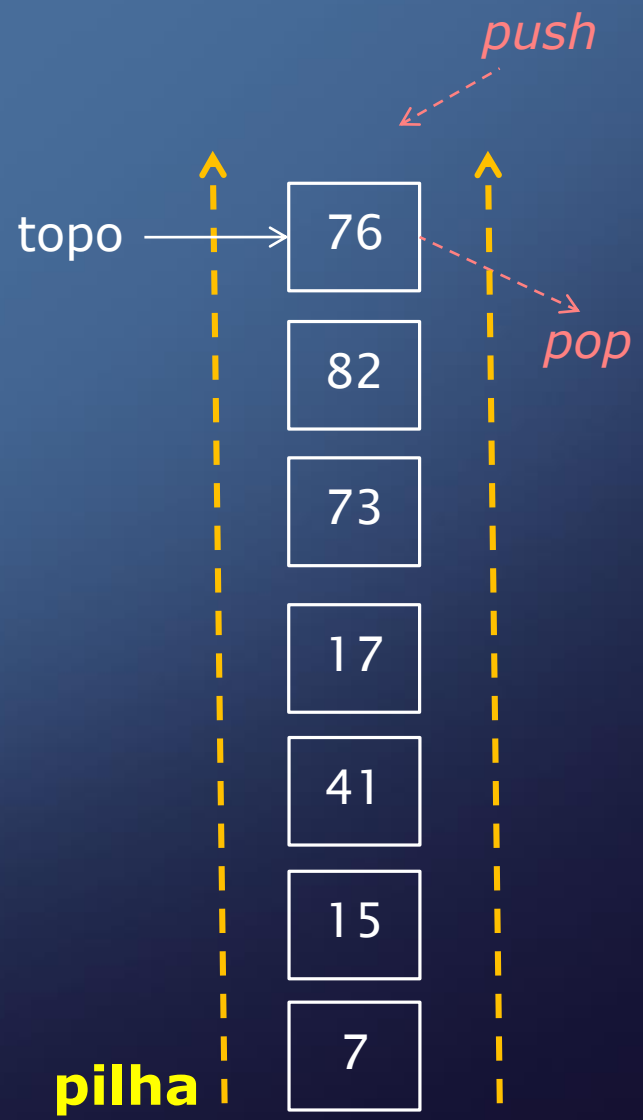
entra na pilha

sai da pilha



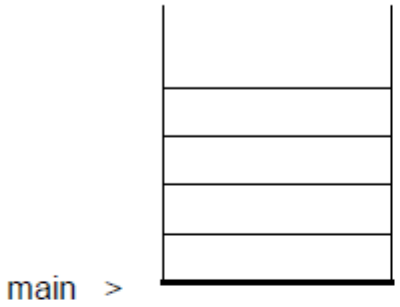
PILHA – CARACTERÍSTICAS

- Pilha é uma estrutura de dados do mesmo tipo que utiliza o algoritmo *last in, first out* – LIFO.
  - ✓ Todo acesso aos elementos é feito a partir do topo.
  - ✓ Inserção e remoção ocorrem no topo da pilha.
- É comum utilizar os termos em inglês para referência às operações de inserção e remoção na pilha: empilhar – *push*; desempilhar – *pop*.
- Aplicações:
  - Análise de expressões matemáticas;
  - Conversão de bases;
  - Hardware da maioria das máquinas (gerenciamento de memória);
  - Compiladores – pilha de execução.

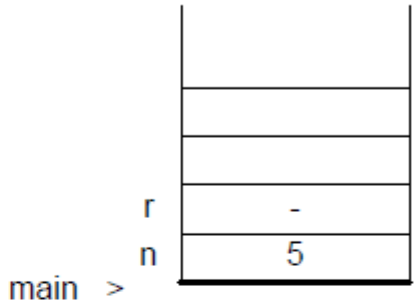


PILHA – ILUSTRAÇÃO DE FUNCIONAMENTO DE UMA PILHA DE EXECUÇÃO

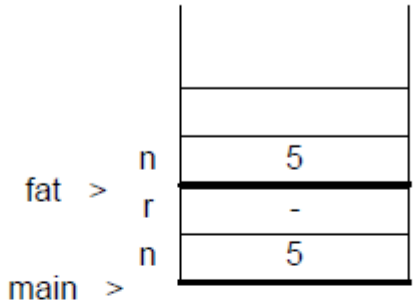
1 - Início do programa: pilha vazia



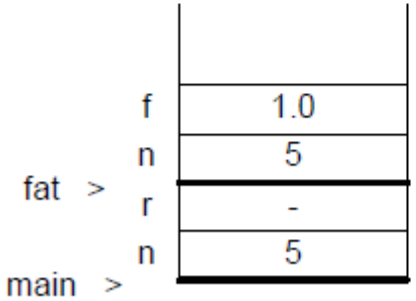
2 - Declaração das variáveis: n, r



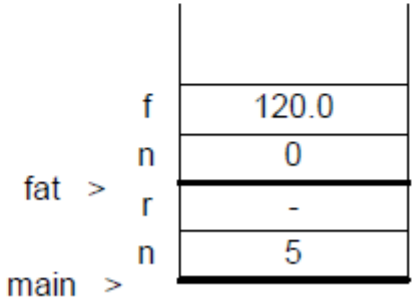
3 - Chamada da função: cópia do parâmetro



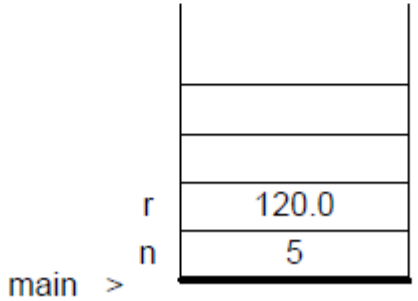
4 - Declaração da variável local: f



5 - Final do laço

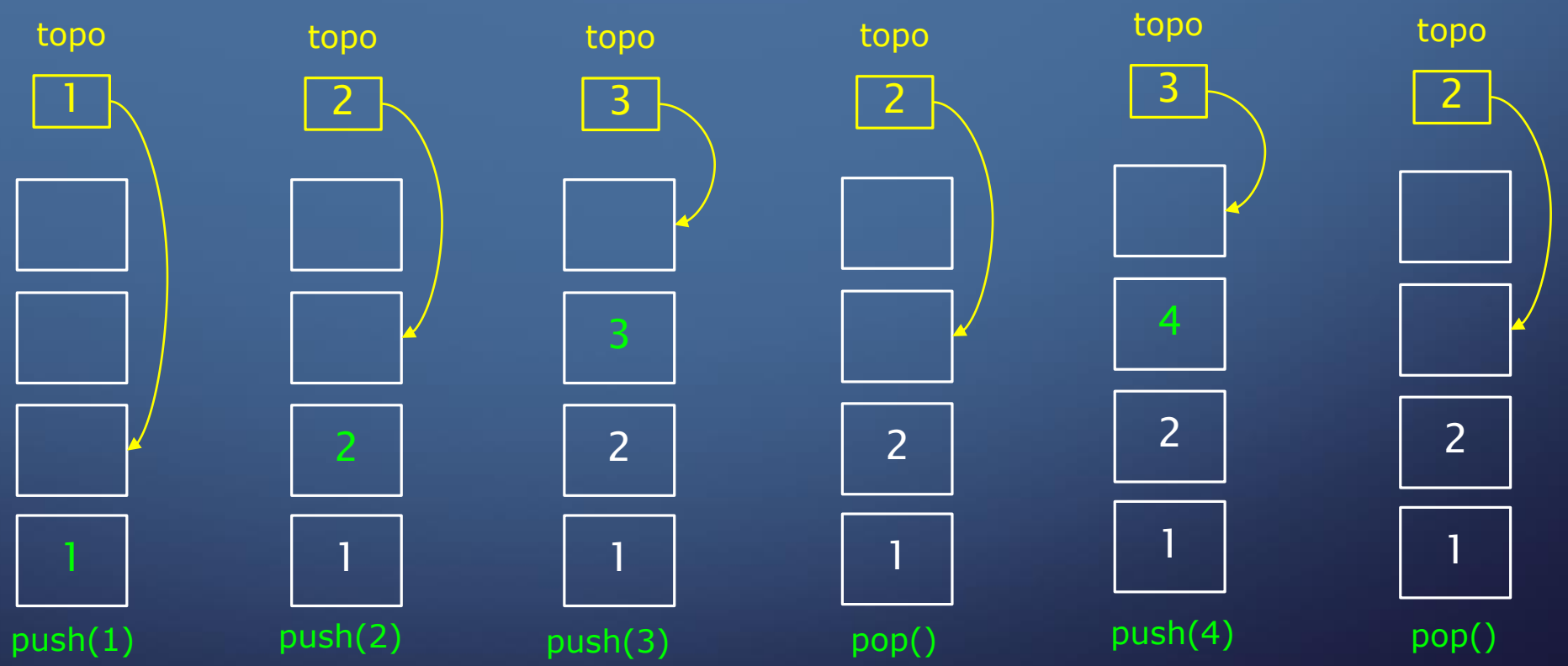


6 - Retorno da função: desempilha



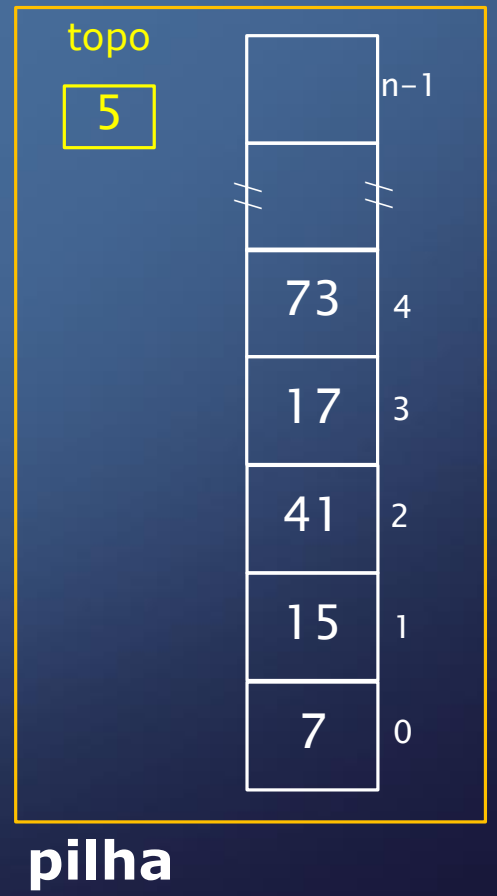
Fonte: retirado de Celes e Rangel, 2002.

PILHA – ILUSTRAÇÃO DE FUNCIONAMENTO *PUSH*() E *POP*()



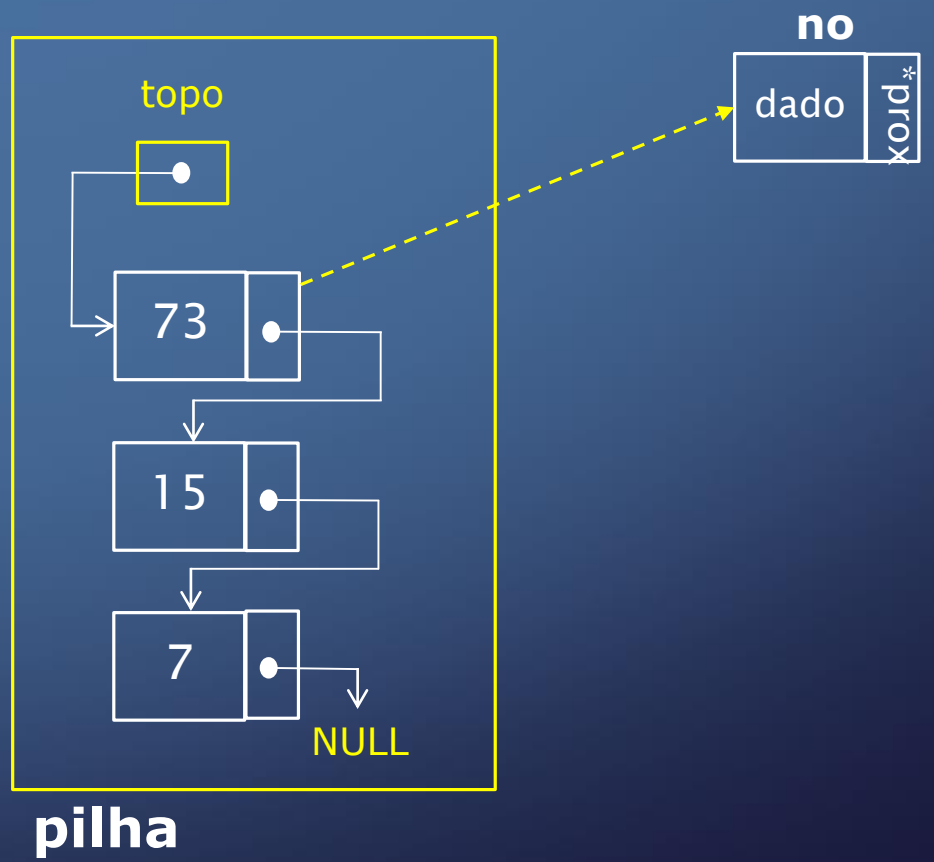
IMPLEMENTAÇÃO DE PILHA QUANTO À ALOCAÇÃO DE MEMÓRIA (1/2)

- Com uso de lista estática (vetor):
  - topo indica a quantidade de elementos na pilha e também a próxima posição a ser ocupada.



IMPLEMENTAÇÃO DE PILHA QUANTO À ALOCAÇÃO DE MEMÓRIA (2/2)

- Com uso de lista dinâmica:



IMPLEMENTAÇÃO DE PILHA USANDO LISTA DINÂMICA: ESTRUTURAS

pilhaDinamica.c | pilhaDinamica.h | main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "pilhaDinamica.h"

struct tipoNo{
    struct elemento dado;
    struct tipoNo *prox;
};
typedef struct tipoNo noPilha;
```

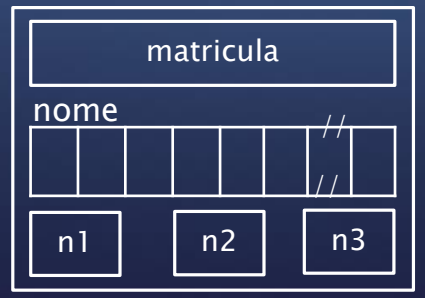
tipoNo ou noPilha

pilhaDinamica.c | pilhaDinamica.h | main.c

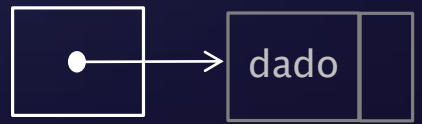
```c
struct elemento{
    int matricula;
    char nome[40];
    float n1, n2, n3;
};

typedef struct tipoNo *Pilha; //PONTEIRO P/ PONTEIRO **,
                              //conforme usado na lista dinâmica
```

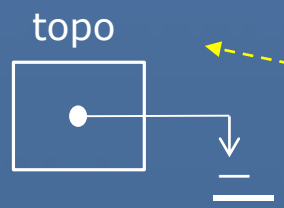
elemento



Pilha



IMPLEMENTAÇÃO: CRIAÇÃO DA PILHA



Programa principal

```
Pilha *topo = criaPilha();
```

```
11  Pilha* criaPilha(){
12      Pilha* top = (Pilha*) malloc(sizeof(Pilha));
13      if(top != NULL)
14          *top = NULL;
15      return top;
16  }
```

DÚVIDAS?

VOCÊ FAZ....

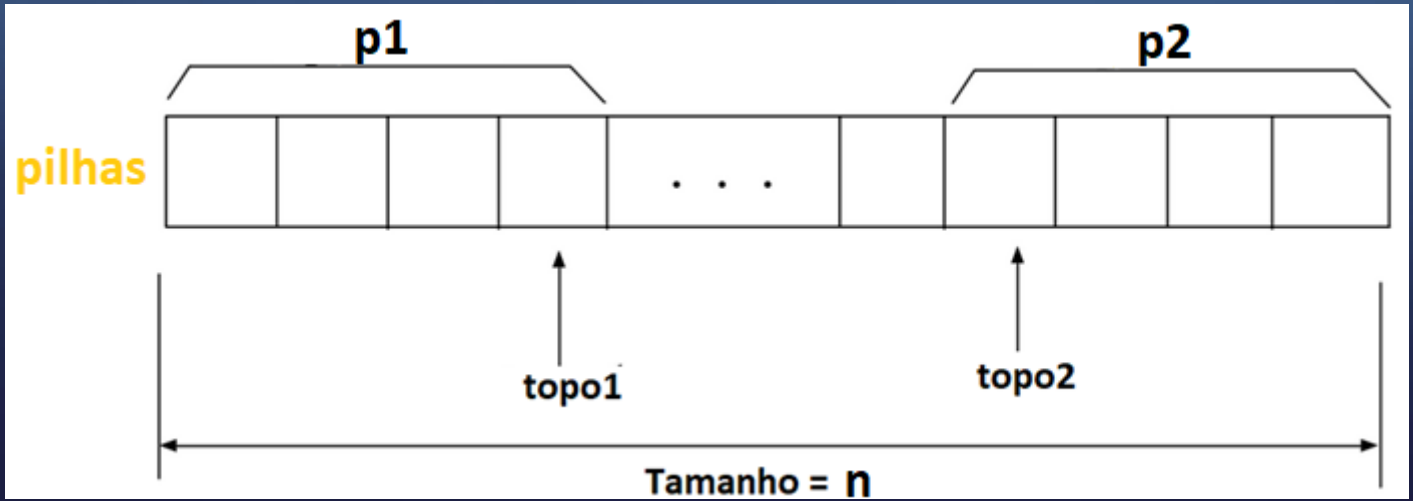
Dado os protótipos, implemente todas as operações de manipulação de pilha dinâmica encadeada com uso de ponteiro especial (**).

```
10  Pilha* criaPilha();
11
12  void apagaPilha(Pilha* topo);
13
14  int push(Pilha* topo, struct elemento aluno);
15
16  void imprimePilha(Pilha* topo);
17
18  int pop(Pilha* topo);
19
20  int consultaTopoPilha(Pilha* topo, struct elemento *aluno);
21
22  int tamPilha(Pilha* topo);
23
24  int pilhaVazia(Pilha* topo);
```

EXERCÍCIO 1 PARA FIXAÇÃO

Implemente um código para controle de inserção e remoção em pilha compartilhada - uso de duas pilhas armazenadas sequencialmente e que compartilham memória de dimensão n.

Verifique *overflow* (estouro da pilha – tentativa de inserir em pilha cheia) e *underflow* (tentativa de remover de pilha vazia).



Lógica:

- "p1" e "p2" são armazenadas no vetor "pilhas".
- p1
  - Cresce da esquerda para a direita.
  - Posição pilhas[1] é o início de p1.
  - Topo da p1 é indicado por topo1.
  - Situação inicial: topo1=0.
- p2
  - cresce da direita para a esquerda.
  - posição pilhas[n] é o início de p2.
  - Topo da p2 é indicado por topo2.
  - Situação inicial: topo2=n+1.

EXERCÍCIO 2 PARA FIXAÇÃO

Explicação sobre o cálculo de expressões pós-fixadas:

Calculadoras da HP (*Hewlett-Packard*) trabalham com expressões pós-fixadas. Para avaliarmos uma expressão como (1-2)*(4+5) podemos digitar 1 2 - 4 5 + *.

O funcionamento dessas calculadoras é muito simples. Cada operando é empilhado numa pilha de valores. Quando se encontra um operador, desempilha-se o número apropriado de operandos (dois para operadores binários e um para operadores unários – exs: fatorial, quadrado, incremento, endereçamento [&]), realiza-se a operação devida e empilha-se o resultado.

Deste modo, na expressão anterior, são empilhados os valores 1 e 2. Quando aparece o operador -, 1 e 2 são desempilhados e o resultado da operação, no caso -1 (= 1 - 2), é colocado no topo da pilha. A seguir, 4 e 5 são empilhados. O operador seguinte, +, desempilha o 4 e o 5 e empilha o resultado da soma, 9. Nesta hora, estão na pilha os dois resultados parciais, -1 na base e 9 no topo. O operador *, então, desempilha os dois e coloca -9 (= -1 * 9) no topo da pilha.

Enunciado do exercício:

implemente uma calculadora pós-fixada usando lista dinâmica. Determine caractere de finalização da execução: usuário continua fazendo operações até desejar parar ou selecionar opção para cálculo de nova expressão. Se preferir, utilize apenas operações binárias. Valide operadores válidos: + - * /.