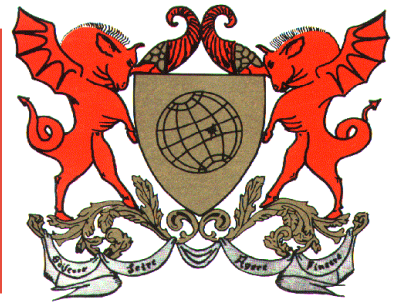


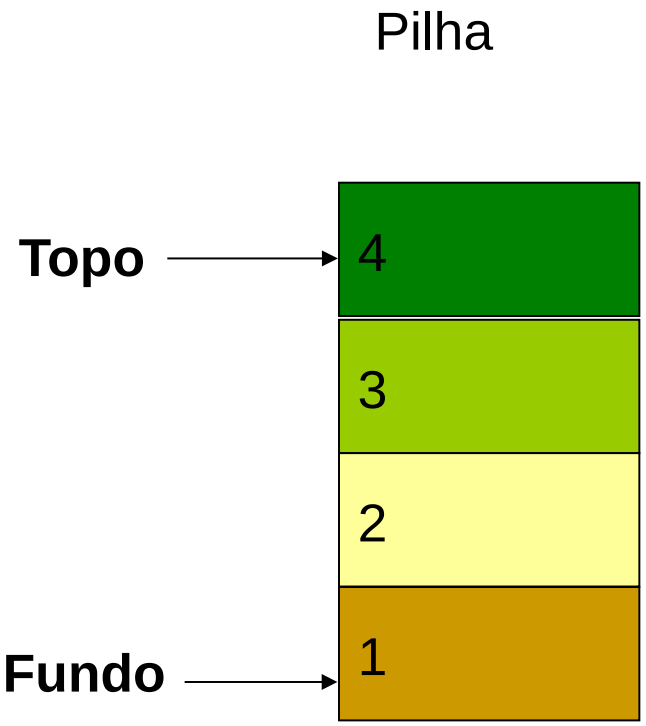
Universidade Federal de Viçosa
Campus de Florestal

Algoritmos e Estruturas de Dados I (CCF 211)

Pilhas (Cap03 – Seção 3.2- Ziviani)

Profa.Thais R. M. Braga Silva
<thais.braga@ufv.br>

O que é uma Pilha?



Pilha

- É uma lista linear em que todas as inserções, retiradas e, geralmente, todos os acessos são feitos em apenas um extremo da lista
- Os itens são colocados uns sobre os outros. O item inserido mais recentemente está no topo e o inserido menos recentemente no fundo.
- O modelo intuitivo é o de um monte de pratos em uma prateleira, sendo conveniente retirar ou adicionar pratos na parte superior.

Propriedades e Aplicações das Pilha

- Propriedade: o último item inserido é o primeiro item que pode ser retirado da lista.
- São chamadas de listas LIFO (*Last in, First Out*).
- Existe uma ordem linear para as pilhas, do “mais recente para o menos recente”.
- É ideal para o processamento de estruturas aninhadas de profundidade imprevisível.
- Uma pilha contém uma sequência de obrigações adiadas. A ordem de remoção garante que as estruturas mais internas serão processadas antes das mais externas.

Propriedades e Aplicações das Pilha

- As pilhas ocorrem em estruturas de natureza recursiva (como árvores). Elas são utilizadas para implementar recursividade.
- Aplicações em estruturas aninhadas:
  - Quando é necessário caminhar em um conjunto de dados e guardar uma lista de coisas a fazer posteriormente
  - O controle de sequências de chamadas de subprogramas
  - A sintaxe de expressões aritméticas

TAD Pilha

- Tipo Abstrato de dados com a seguinte característica:

  O **último** elemento a ser inserido é o **primeiro** a ser retirado/ removido

  (LIFO – *Last in First Out*)

- Analogia: pilha de pratos, livros, etc.

- Usos: Chamada de subprogramas, avalição de expressões aritméticas, etc.

TAD Pilha

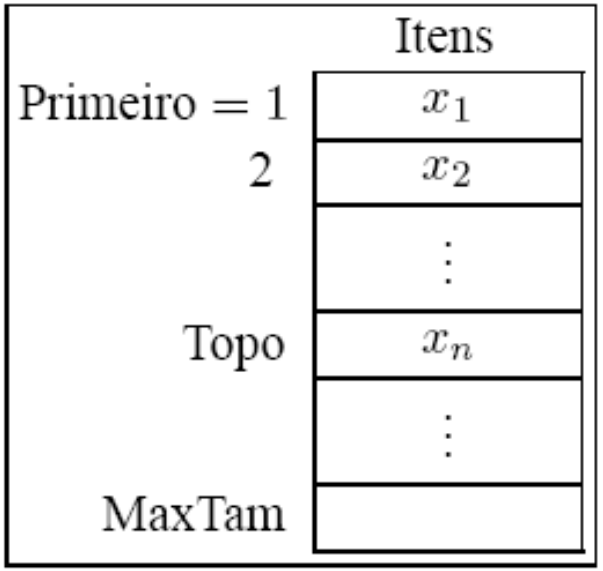
- Operações:
  - 1. FPVazia(Pilha). Faz a pilha ficar vazia.
  - 2. PEhVazia(Pilha). Retorna *true* se a pilha está vazia; caso contrário, retorna *false*.
  - 3. PEmpilha(Pilha, x). Insere o item x no topo da pilha.
  - 4. PDesempilha(Pilha, x). Retorna o item x no topo da pilha, retirando-o da pilha.
  - 5. PTamanho(Pilha). Esta função retorna o número de itens da pilha

- **As duas representações mais utilizadas para representar pilhas são as implementações por meio de arranjos e de apontadores**

Implementação de Pilhas através de Arranjos

- Os itens da pilha são armazenados em posições contíguas de memória.

- Como as inserções e as retiradas ocorrem no topo da pilha, um campo chamado Topo é utilizado para controlar a posição do item no topo da pilha.



Estrutura de Dados de Pilha através de Arranjos

```
#define MaxTam 1000

typedef int Apontador;
typedef int TChave;
typedef struct {
  TChave Chave;
  /* outros componentes */
} TItem;
typedef struct {
  TItem vItem[MaxTam];
  Apontador iTopo;
} TPilha;
```

Estrutura de Dados de Pilha através de Arranjos

```
void FPVazia(TPilha* pPilha);

int PEhVazia(TPilha* pPilha);

int PEmpilha(TPilha* pPilha,TItem*  pItem);

int PDesempilha(TPilha* pPilha,TItem*  pItem);

int PTamanho(TipoPilha* pPilha);
```

Operações sobre Pilhas usando Arranjos

```
void FPVazia(TPilha* pPilha)
{
  pPilha->iTopo = 0;
} /* FPVazia */

int PEhVazia(TPilha* pPilha)
{
  return (pPilha->iTopo == 0);
} /* PEhVazia */
```

Operações sobre Pilhas usando Arranjos

```
int PEmpilha(TPilha* pPilha,
             TItem*  pItem)
{
  if (pPilha->iTopo == MaxTam)
    return 0;

  pPilha->vItem[pPilha->iTopo] = *pItem;
  pPilha->iTopo++;
  return 1;
} /* PEmpilha */
```

Operações sobre Pilhas usando Arranjos

```
int PDesempilha(TPilha* pPilha,
                TItem*  pItem)
{
  if (PEhVazia(pPilha))
    return 0;

  pPilha->iTopo--;
  *pItem = pPilha->vItem[pPilha->iTopo];
  return 1;
} /* PDesempilha */
```

Operações sobre Pilhas usando Arranjos

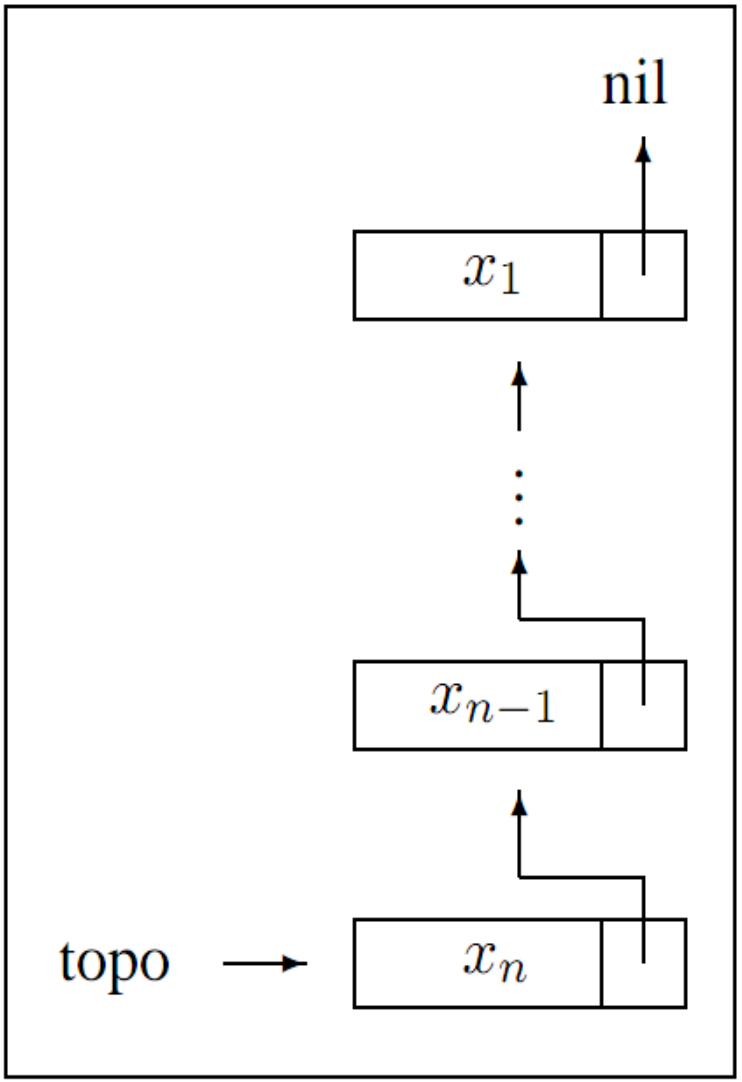
```
int PTamanho(TipoPilha* pPilha)
{
  return (pPilha->iTopo);
} /* Tamanho */
```

Implementação de Pilhas por meio de Apontadores

- **Não há necessidade do uso de uma célula cabeça no topo para a implementação das operações empilha e desempilha.**

Para desempilhar o item x_n basta desligar a célula que contém x_n e a célula que contém x_{n-1} passa a ser a célula de topo.

- **Para empilhar um novo item, basta fazer a operação contrária, criando uma nova célula para receber o novo item.**



Estrutura da Pilha Usando Apontadores

- **O campo Tamanho evita a contagem do número de itens na função Tamanho**
- **Cada célula de uma pilha contém um item da pilha e um apontador para outra célula**
- **O registro TPilha contém um apontador para o topo da pilha**

Estrutura da Pilha Usando Apontadores

```
typedef int TChave;
typedef struct {
  TChave Chave;
  /* --- outros componentes --- */
} TItem;

typedef struct Celula* Apontador;
typedef struct Celula {
  TItem Item;
  struct Celula* pProx; // Apontador pProx
} TCelula;

typedef struct {
  Apontador pTopo;
  int iTamanho;
} TPilha;
```

Estrutura da Pilha Usando Apontadores

```
void FPVazia(TPilha* pPilha);

int PEhVazia(TPilha* pPilha);

int PEmpilha(TPilha* pPilha,TItem*  pItem);

int PDesempilha(TPilha* pPilha,TItem*  pItem);

int PTamanho(TipoPilha* pPilha);
```

Operações sobre Pilhas usando Apontadores (sem cabeça)

```
void FPVazia(TPilha* pPilha)
{
  pPilha->pTopo  = NULL;
  pPilha->iTamanho = 0;
} /* FPVazia */

int PEhVazia(TPilha* pPilha)
{
  return (pPilha->pTopo == NULL);
} /* PEhVazia */
```

Operações sobre Pilhas usando Apontadores (sem cabeça)

```
int PEmpilha(TPilha* pPilha,
             TItem*  pItem)
{
  Apontador pNovo;
  pNovo = (Apontador) malloc(sizeof(TCelula));
  if (pNovo == NULL)
    return 0;

  pNovo->Item = *pItem;
  pNovo->pProx = pPilha->pTopo;
  pPilha->pTopo = pNovo;
  pPilha->iTamanho++;
  return 1;
} /* PEmpilha */
```

Operações sobre Pilhas usando Apontadores (sem cabeça)

```
int PDesempilha(TPilha* pPilha,
                TItem*  pItem)
{
  Apontador pAux; /* celula a ser removida */
  if (PEhVazia(pPilha))
    return 0;

  pAux = pPilha->pTopo;
  pPilha->pTopo = pAux->pProx;
  *pItem = pAux->Item;
  free(pAux);
  pPilha->iTamanho--;
  return 1;
} /* PDesempilha */
```

Operações sobre Pilhas usando Apontadores (sem cabeça)

```
int PTamanho(TipoPilha* pPilha)
{
  return (pPilha->iTamanho);
} /* PTamanho */
```

Exercícios

- **Utilize uma pilha para inverter uma string, ou seja, para escrever um texto em sentido inverso**

- **Implemente um programa que utilize uma pilha para avaliar se o "aninhamento" de expressões delimitadas por "( )" está correto. Exemplos:**
  - a = b + (c-d) * (e-f); //correto
  - s = t + u / (v * (w + y))); // incorreto