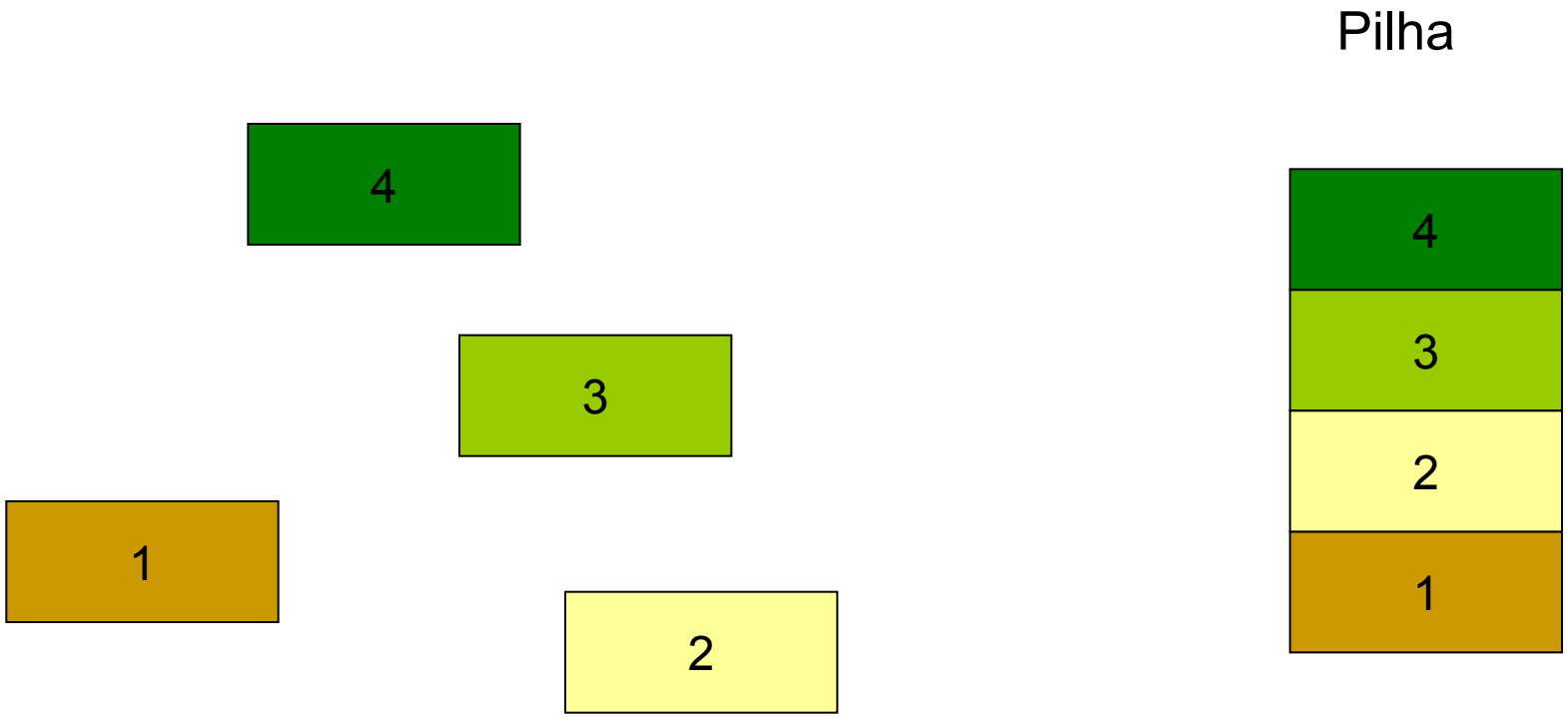
Pilha

David Menotti

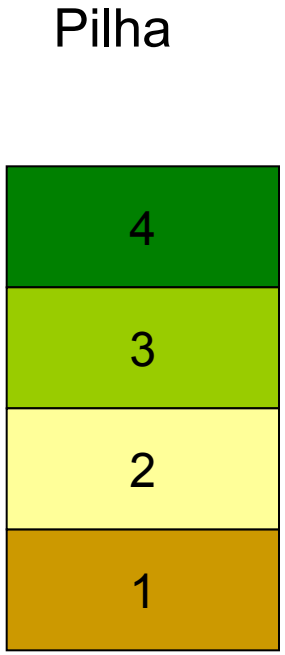
Algoritmos e Estruturas de Dados II

DInf – UFPR

O que é uma pilha?



O que é uma pilha?



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.

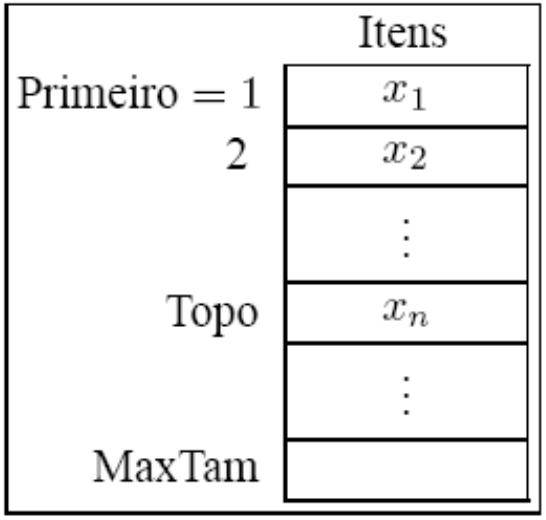
TAD Pilha

- 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

- **Existem várias opções de estruturas de dados que podem ser usadas para representar pilhas.**
- **As duas representações mais utilizadas 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;
```

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

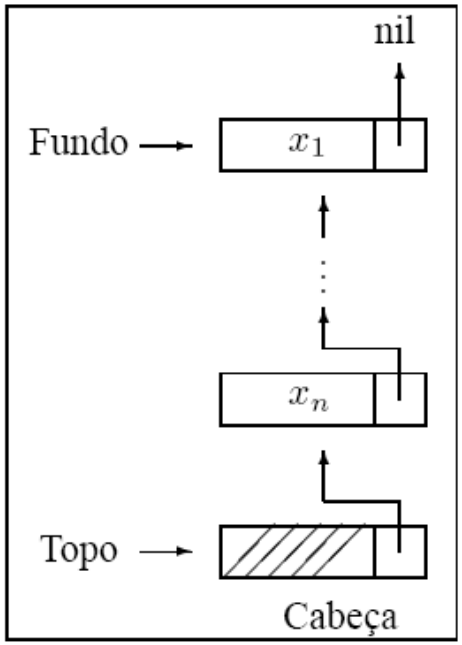
```
int PDesempilha(TPilha* pPilha,
                TItem*  pItem)
{
  if (PEhVazia(pPilha))
    return 0;

  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

- Há uma célula cabeça no topo para facilitar a implementação das operações empilha e desempilha quando a pilha está vazia.
- Para desempilhar o item x_n basta desligar a célula cabeça da lista e a célula que contém x_n passa a ser a célula cabeça.
- Para empilhar um novo item, basta fazer a operação contrária, criando uma nova célula cabeça e colocando o novo item na antiga.



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 (célula cabeça) e um apontador para o fundo 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 pFundo;
  Apontador pTopo;
  int iTamanho;
} TPilha;
```

Operações sobre Pilhas usando Apontadores (com cabeça)

```
void FPVazia(TPilha* pPilha)
{
  pPilha->pTopo  = (Apontador)malloc(sizeof(TCelula));
  pPilha->pFundo = pPilha->pTopo;
  pPilha->pTopo->pProx = NULL;
  pPilha->iTamanho = 0;
} /* FPVazia */

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

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 (com cabeça)

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

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

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

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 e com cabeça)

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

Exemplo de Uso Pilhas
Editor de Textos (ET)

- Vamos escrever um Editor de Texto (ET) que aceite os comandos:
  - Cancela caracter
  - Cancela linha
  - Imprime linha

- O ET deverá ler um caractere de cada vez do texto de entrada e produzir a impressão linha a linha, cada linha contendo no máximo 70 caracteres de impressão.

- O ET deverá utilizar o tipo abstrato de dados Pilha definido anteriormente, implementado por meio de arranjo.

Exemplo de Uso Pilhas
Editor de Textos (ET)

- **"#": cancelar caractere anterior na linha sendo editada.**
  Ex.: UFM##FOB#P DCC##ECOM!

- **"\": cancela todos os caracteres anteriores na linha sendo editada.**

- **"*": salta a linha.**
- **"!": Imprime os caracteres que pertencem à linha sendo editada, iniciando uma nova linha de impressão a partir do caractere imediatamente seguinte ao caractere salta-linha. Ex: DECOM*UFOP*!**
  - DECOM
  - UFOP.

Sugestão de Texto para Testar o ET

```
Este et# um teste para o ET, o
extraterrestre em C.*Acabamos de
testar a capacidade de o ET saltar de
linha, utilizando seus poderes extras
(cuidado, pois agora vamos estourar a
capacidade máxima da linha de
impressão, que é de 70  caracteres.)*O
k#cut#rso dh#e Estruturas de Dados et#
h#um cuu#rsh#o #x# x?*!#?!#+.* Como
et# bom n#nt#ao### r#ess#tt#ar
mb#aa#triz#cull#ado nn#x#ele!\ Sera
que este funciona\\\? O sinal? não###
deve ficar! ~
```

ET - Implementação

- Este programa utiliza um tipo abstrato de dados sem conhecer detalhes de sua implementação.
- A implementação do TAD Pilha que utiliza arranjo pode ser substituída pela implementação que utiliza apontadores sem causar impacto no programa

ET - Implementação

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

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

int PEmpilha(TipoPilha* pPilha,TipoItem* pItem)
{
  if (pPilha->iTopo == MaxTam)
    return 0;
  else
  {
    pPilha->iTopo++;
    pPilha->Item[Pilha->iTopo - 1] = *pItem;
  }
  return 1;
} /* PEmpilha */
```

ET - Implementação

```
int PDesempilha(TipoPilha* pPilha, TipoItem* pItem)
{
  if (PEhVazia(pPilha))
    return 0
  else
  {
    *pItem = pPilha->Item[pPilha->iTopo - 1];
    pPilha->iTopo--;
  }
  return 1;
} /* PDesempilha */

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

ET - Implementação

```
int main(int argc, char* argv[])
{
  TipoPilha Pilha;
  TipoItem x;
  FPVazia(&Pilha);
  x.Chave = getchar();
  while (x.Chave != MarcaEof)
  {
    if (x.Chave == CancelaCarater)
    { if (!PEhVazia(&Pilha)) PDesempilha(&Pilha, &x);}
    else if (x.Chave == CancelaLinha) FPVazia(&Pilha);
    else if (x.Chave == SaltaLinha)   PImprime(&Pilha);
    else
    { if (PTamanho(Pilha) == MaxTam) PImprime(&Pilha);
      PEmpilha(&Pilha, &x); }
    x.Chave = getchar();
  }
  if (!PEhVazia(&Pilha)) PImprime(&Pilha);
  return 0;
} /* ET */
```

ET - Implementação

```
void PImprime(TipoPilha* pPilha)
{
  TipoPilha Pilhaux;
  TipoItem x;
  FPVazia(&Pilhaux);
  while (!PEhVazia(pPilha))
  {
    PDesempilha(pPilha, &x); PEmpilha(&Pilhaux,&x);
  }
  while (!PEhVazia(&Pilhaux))
  {
    PDesempilha(&Pilhaux, &x); putchar(x.Chave);
  }
  putchar('\n');
} /* Imprime */
```