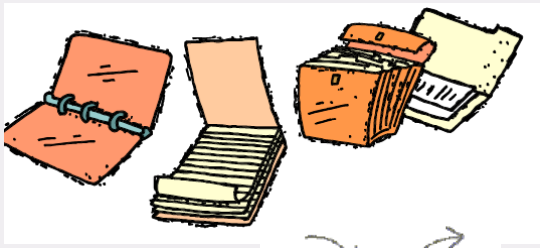
Agenda

Unidade 3: Listas Lineares

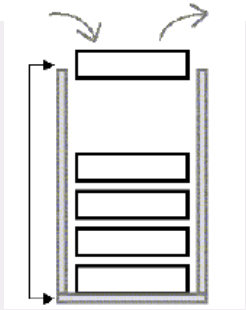
1. Listas lineares
2. **Pilhas**
3. Filas
4. Implementações

Listas, Pilhas e Filas

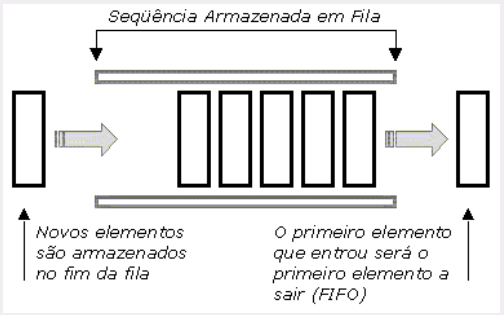
- Listas
  - Inserção: em qualquer posição
  - Remoção: em qualquer posição



- Pilhas
  - Inserção: Topo (primeira posição na lista)
  - Remoção: Topo (primeira posição na lista)

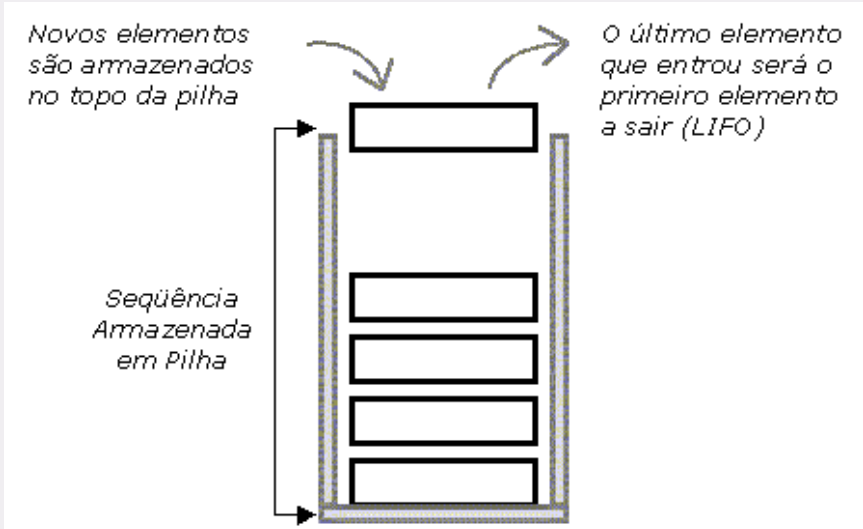


- Filas
  - Inserção: última posição na lista
  - Remoção: primeira posição na lista



Pilhas

- É uma **lista linear** em que todas as operações de inserção e remoção são feitas por um único extremo denominado topo, ou seja, as pilhas são estruturas baseadas no princípio LIFO *(last in, first out)*.
  - O último elemento a ser inserido é o primeiro a ser retirado (LIFO)
- Analogia: pilha de pratos, livros, etc
- Usos: chamadas de subprogramas



Pilhas

- Existe uma ordem linear para pilhas, do “mais recente para o menos recente”.
- É ideal para 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.

Pilhas

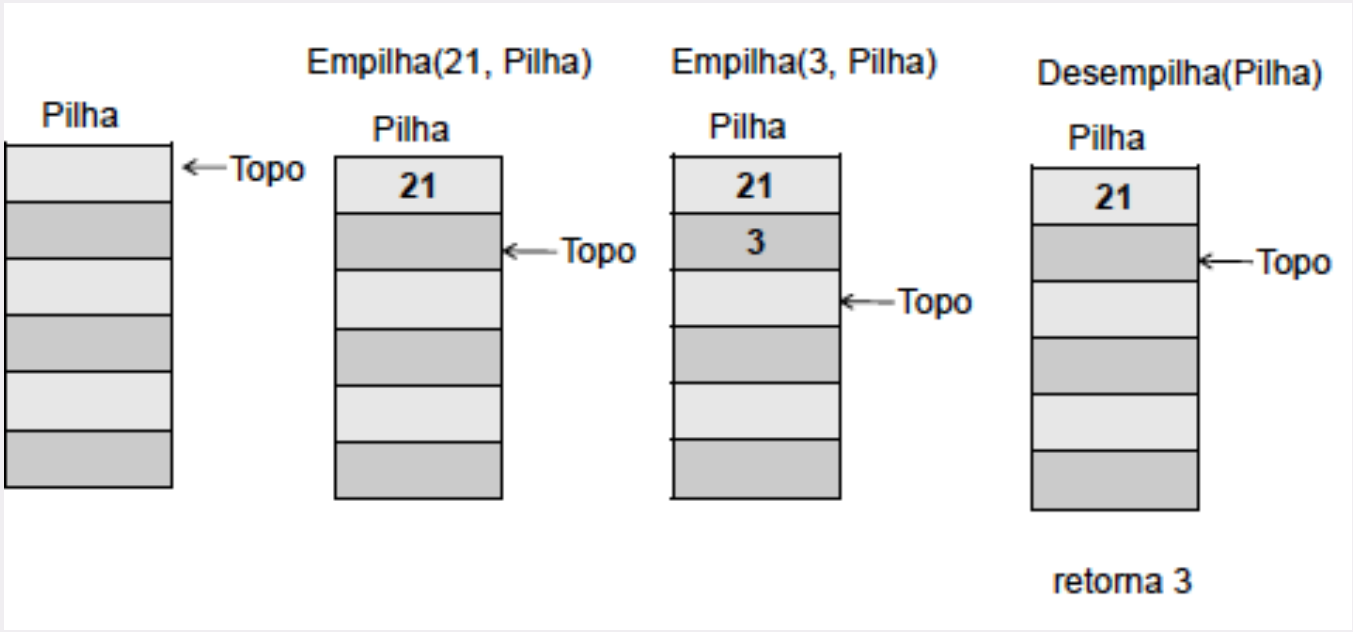
- Aplicações em estruturas aninhadas:
  - Quando é necessário caminhar em um conjunto de dados e guardar uma lista de coisas a fazer posteriormente.
  - O controle de seqüencias de chamadas de subprogramas.
  - A sintaxe de expressões aritméticas.
- As pilhas ocorrem em estruturas de natureza recursiva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

TAD Pilhas

- Conjunto de operações:
  1. FPVazia(Pilha). Faz a pilha ficar vazia.
  2. Vazia(Pilha). Retorna true se a pilha está vazia; caso contrário, retorna false.
  3. Empilha(x, Pilha). Insere o item x no topo da pilha. (Operação *PUSH)*
  4. Desempilha(Pilha, x). Retorna o item x no topo da pilha, retirando-o da pilha. (Operação POP)
  5. Tamanho(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 por meio de Arranjos

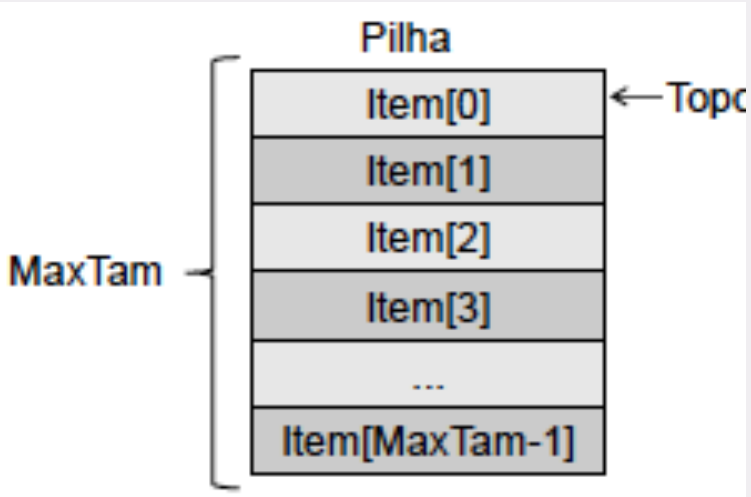
- Os itens da pilha são armazenados em posições contíguas de memória.
- Como as inserções e as retiradas ocorrem no topo da pilha, um cursor chamado Topo é utilizado para controlar a posição do item no topo da pilha.



Estrutura da Pilha Usando Arranjo

- Os itens são armazenados em um array do tamanho da pilha.
- O outro campo do mesmo registro contém um apontador para o item no topo da pilha.
- A constante MaxTam define o tamanho máximo permitido para a pilha.

```
#define MAXTAM 1000
typedef int TipoApontador;
typedef int TipoChave;
typedef struct {
  TipoChave Chave;
  /* ---- outros componentes ---- */
} TipoItem;
typedef struct {
  TipoItem Item[MAXTAM];
  TipoApontador Topo;
} TipoPilha;
```

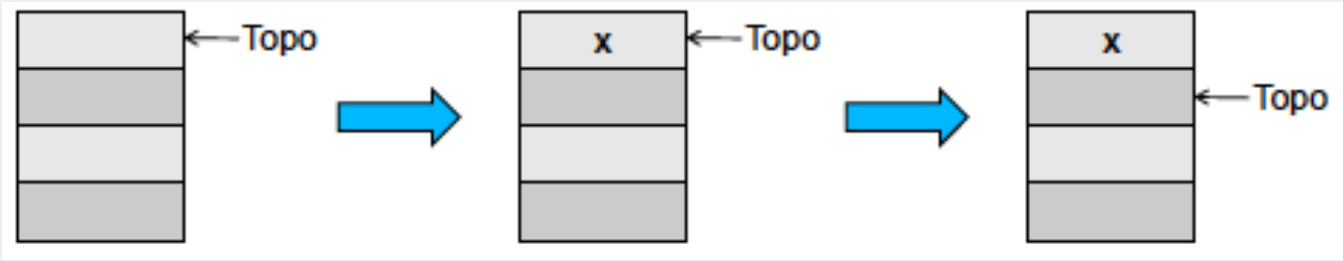


Operações sobre Pilhas Usando Arranjos

```
void FPVazia(TipoPilha *Pilha)
{
  Pilha->Topo = 0;
}  /* FPVazia */



int Vazia(TipoPilha *Pilha)
{
  return (Pilha->Topo == 0);
}  /* Vazia */
```

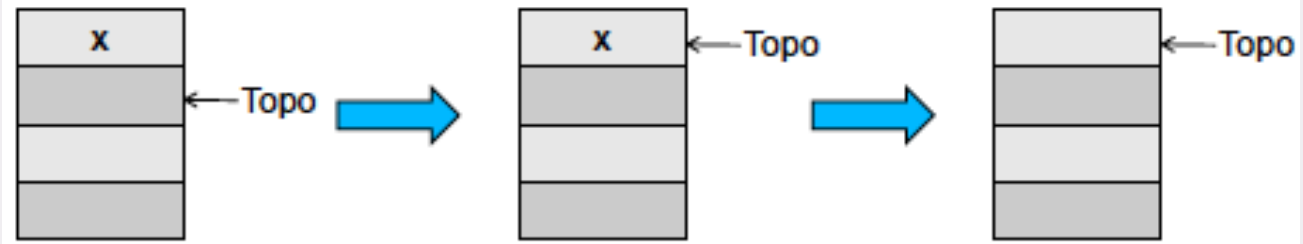
Operações sobre Pilhas Usando Arranjos

```
void Empilha(TipoItem x, TipoPilha *Pilha)
{
  if (Pilha->Topo == MaxTam)
      printf("Erro: pilha está cheia\n");
  else {
      Pilha->Item[Pilha->Topo] = x;
      Pilha->Topo++;
  }
}
```



Operações sobre Pilhas Usando Arranjos

```
TipoItem Desempilha(TipoPilha *Pilha)
{
  if (Vazia(Pilha))
      printf("Erro: a pilha está vazia\n");
  else {
       Pilha->Topo--;
      return Pilha->Item[Pilha->Topo];
  }
}
```

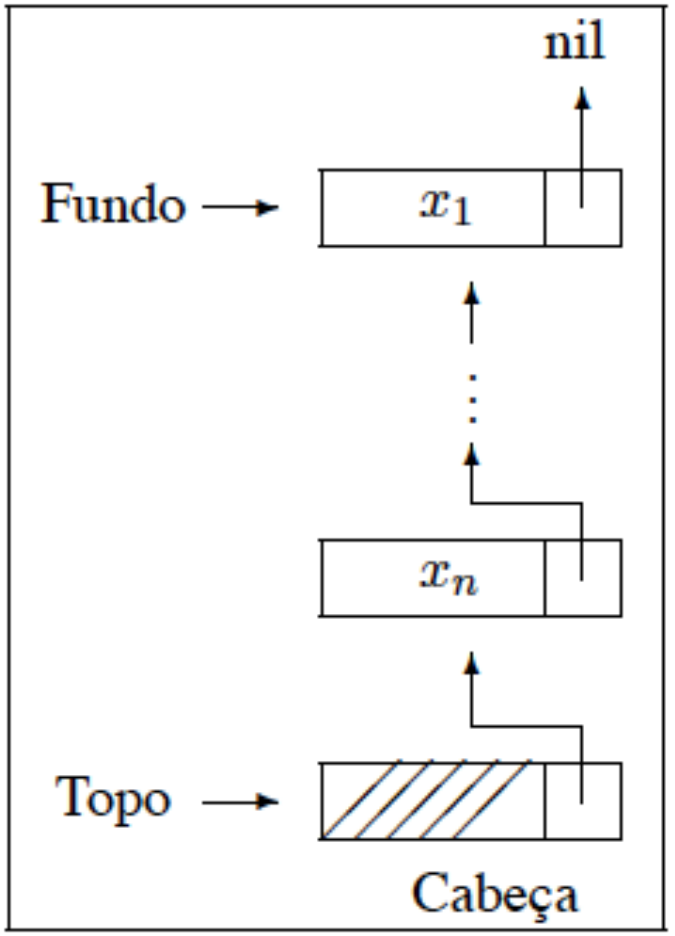


Operações sobre Pilhas Usando Arranjos

```
int Tamanho(TipoPilha *Pilha)
{
   return Pilha->Topo;
}
```

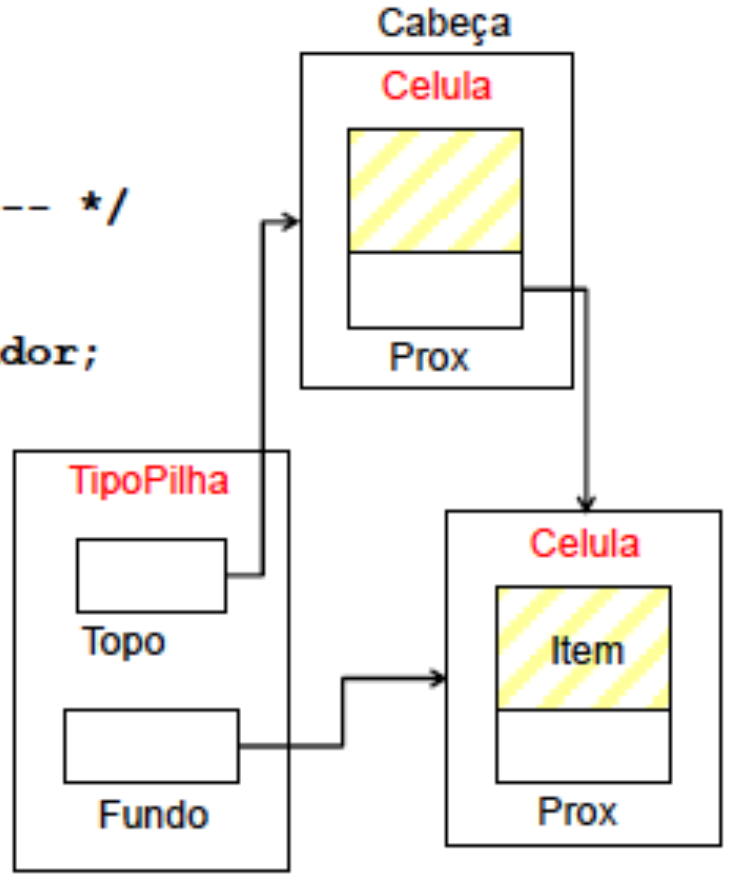
Impelementação de Pilhas por meio de Apontadores

- Há uma célula cabeça no topo para facilitar a implementação das operações empilha e desempilha quando a pilha está vazia.
- Desempilha: desliga a célula cabeça da lista e a célula que contém o primeiro item, passa a ser a célula cabeça.
- Empilha: Cria uma nova célula cabeça e adiciona o item a ser empilhado na antiga célula cabeça



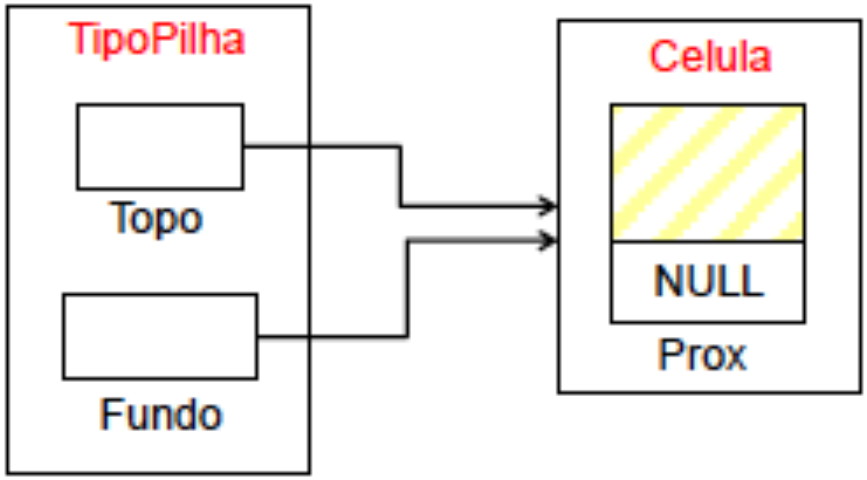
Estrutura da Pilha Usando Apontadores

```
typedef int TipoChave;
typedef struct {
  int Chave;
  /* --- outros componentes --- */
} TipoItem;

typedef struct Celula *Apontador;

typedef struct Celula {
  TipoItem Item;
  Apontador Prox;
} Celula;

typedef struct {
  Apontador Fundo, Topo;
  int Tamanho;
} TipoPilha;
```



Operações sobre Pilhas Usando Apontadores

```
void FPVazia(TipoPilha *Pilha)
{
  Pilha->Topo = (Apontador) malloc(sizeof(Celula));
  Pilha->Fundo = Pilha->Topo;
  Pilha->Topo->Prox = NULL;
  Pilha->Tamanho = 0;
}  /* FPVazia */
```

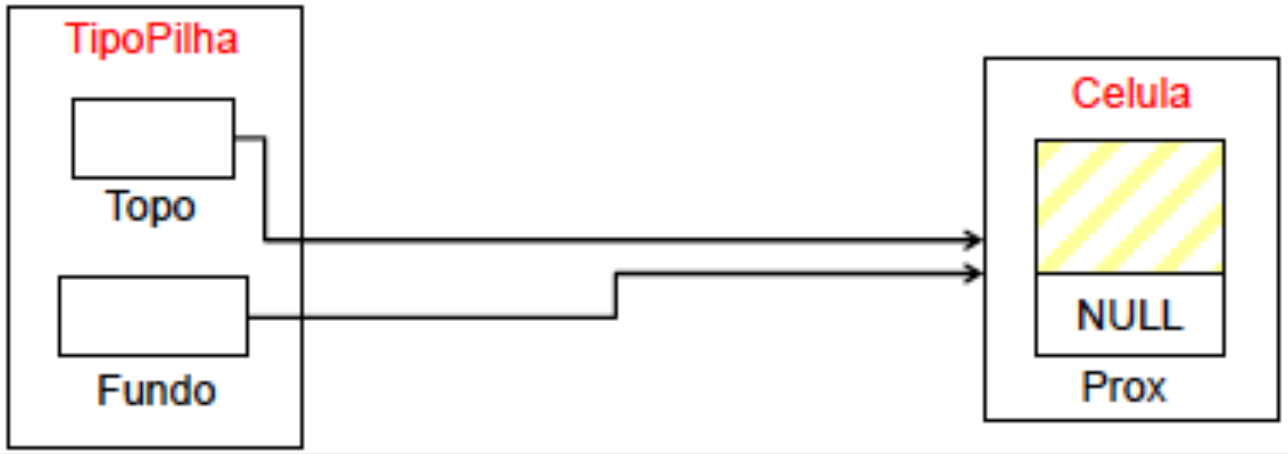


Operações sobre Pilhas Usando Apontadores

```
int Vazia(TipoPilha *Pilha)
{
   return (Pilha->Topo == Pilha->Fundo);
}  /* Vazia */
```

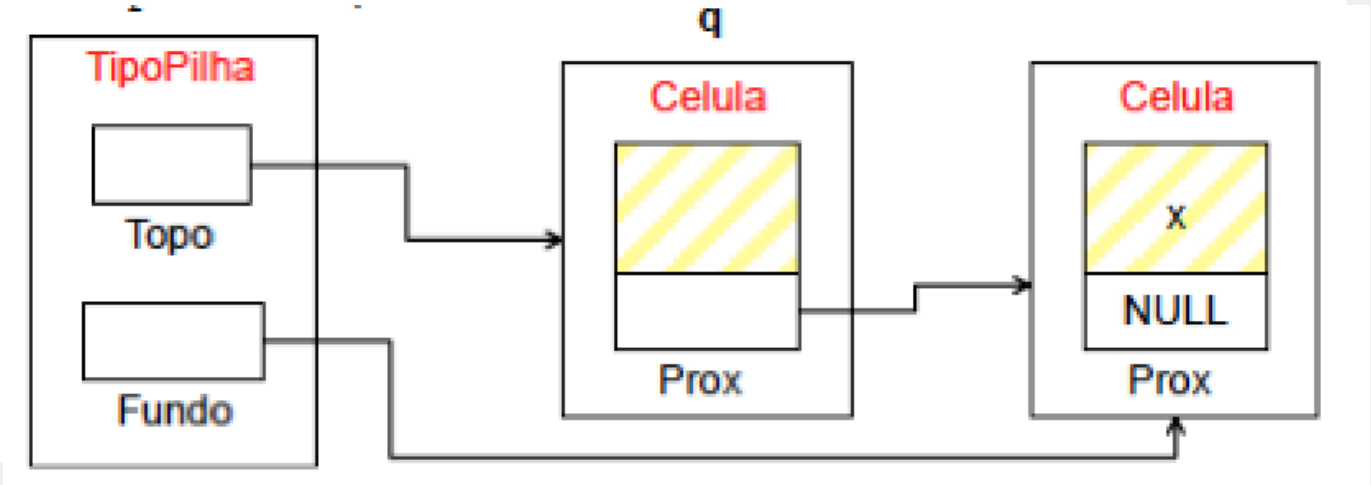
Operações sobre Pilhas Usando Apontadores

```
void Empilha(TipoItem x, TipoPilha *Pilha) {
  Apontador Aux;
  Aux = (Apontador) malloc(sizeof(Celula));
  Pilha->Topo->Item = x;
  Aux->Prox = Pilha->Topo;
  Pilha->Topo = Aux;
  Pilha->Tamanho++;
}
```



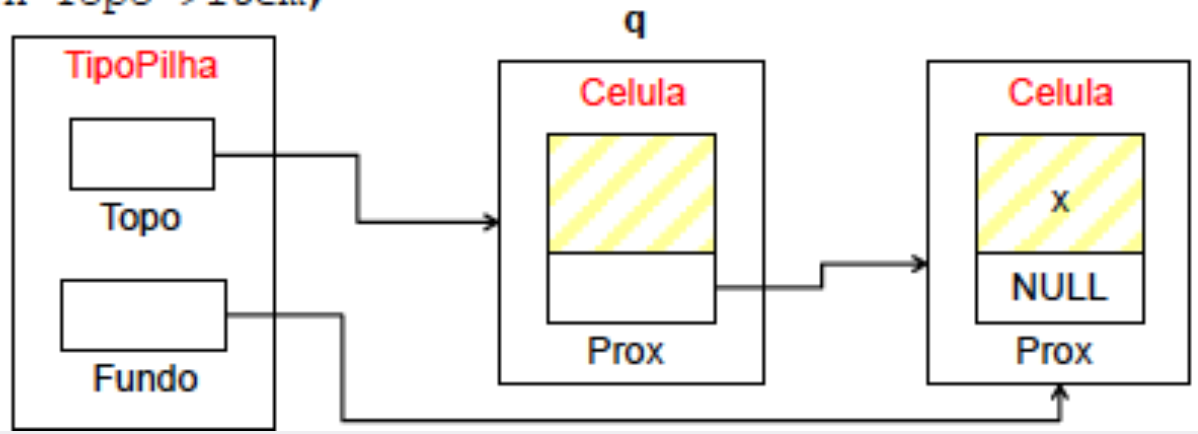
Operações sobre Pilhas Usando Apontadores

```
void Empilha(TipoItem x, TipoPilha *Pilha) {
  Apontador Aux;
  Aux = (Apontador) malloc(sizeof(Celula));
  Pilha->Topo->Item = x;
  Aux->Prox = Pilha->Topo;
  Pilha->Topo = Aux;
  Pilha->Tamanho++;
}
```



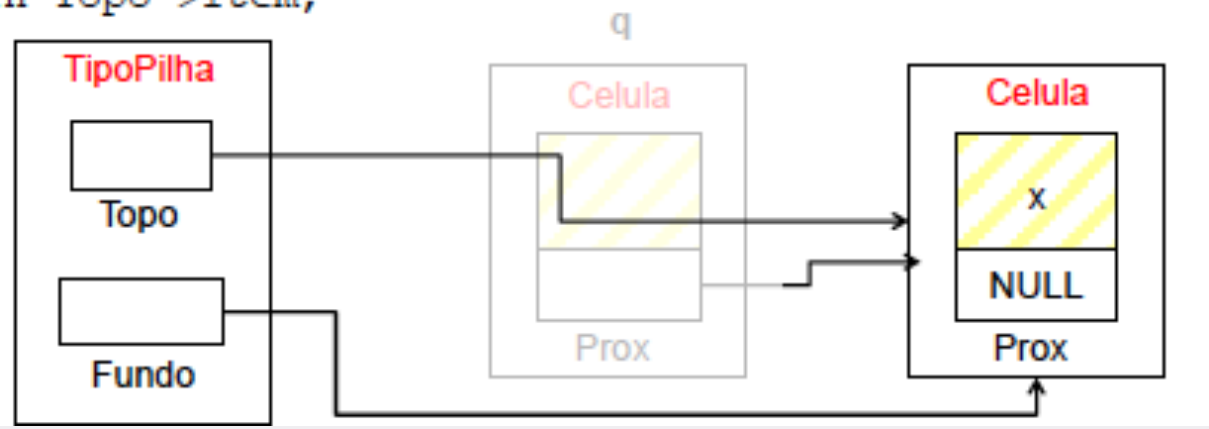
Operações sobre Pilhas Usando Apontadores

```
TipoItem Desempilha(TipoPilha *Pilha){
  Apontador q;
  if (Vazia(Pilha)) {
      printf("Erro: pilha vazia\n"); ERRO;
  }
  q = Pilha->Topo;
  Pilha->Topo = q->Prox;
  free(q);
  Pilha->Tamanho--;
  return Topo->Item;
}
```



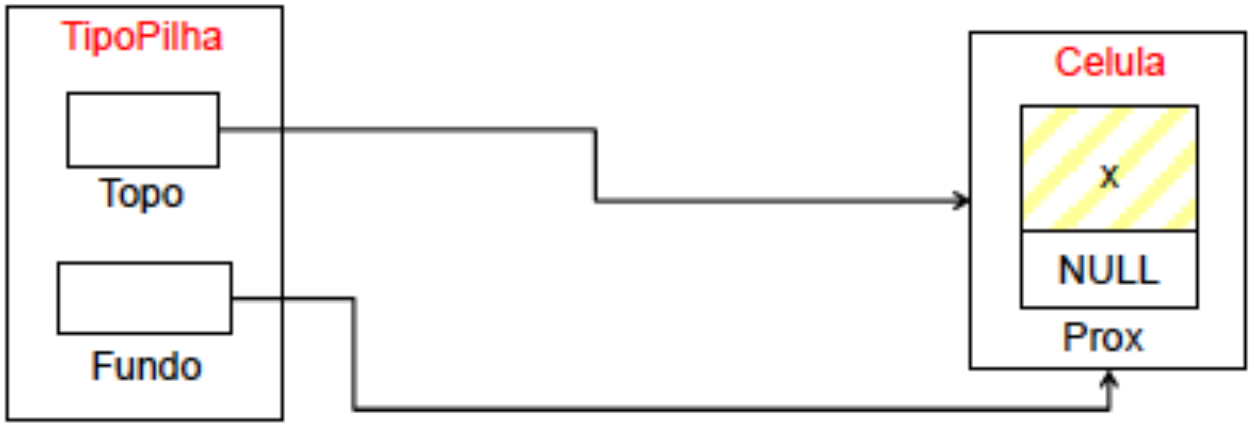
Operações sobre Pilhas Usando Apontadores

```
TipoItem Desempilha(TipoPilha *Pilha){
  Apontador q;
  if (Vazia(Pilha)) {
      printf("Erro: pilha vazia\n"); ERRO;
  }
  q = Pilha->Topo;
  Pilha->Topo = q->Prox;
  free(q);
  Pilha->Tamanho--;
  return Topo->Item;
}
```



Operações sobre Pilhas Usando Apontadores

```
TipoItem Desempilha(TipoPilha *Pilha){
  Apontador q;
  if (Vazia(Pilha)) {
      printf("Erro: pilha vazia\n"); ERRO;
  }
  q = Pilha->Topo;
  Pilha->Topo = q->Prox;
  free(q);
  Pilha->Tamanho--;
  return Topo->Item;
}
```

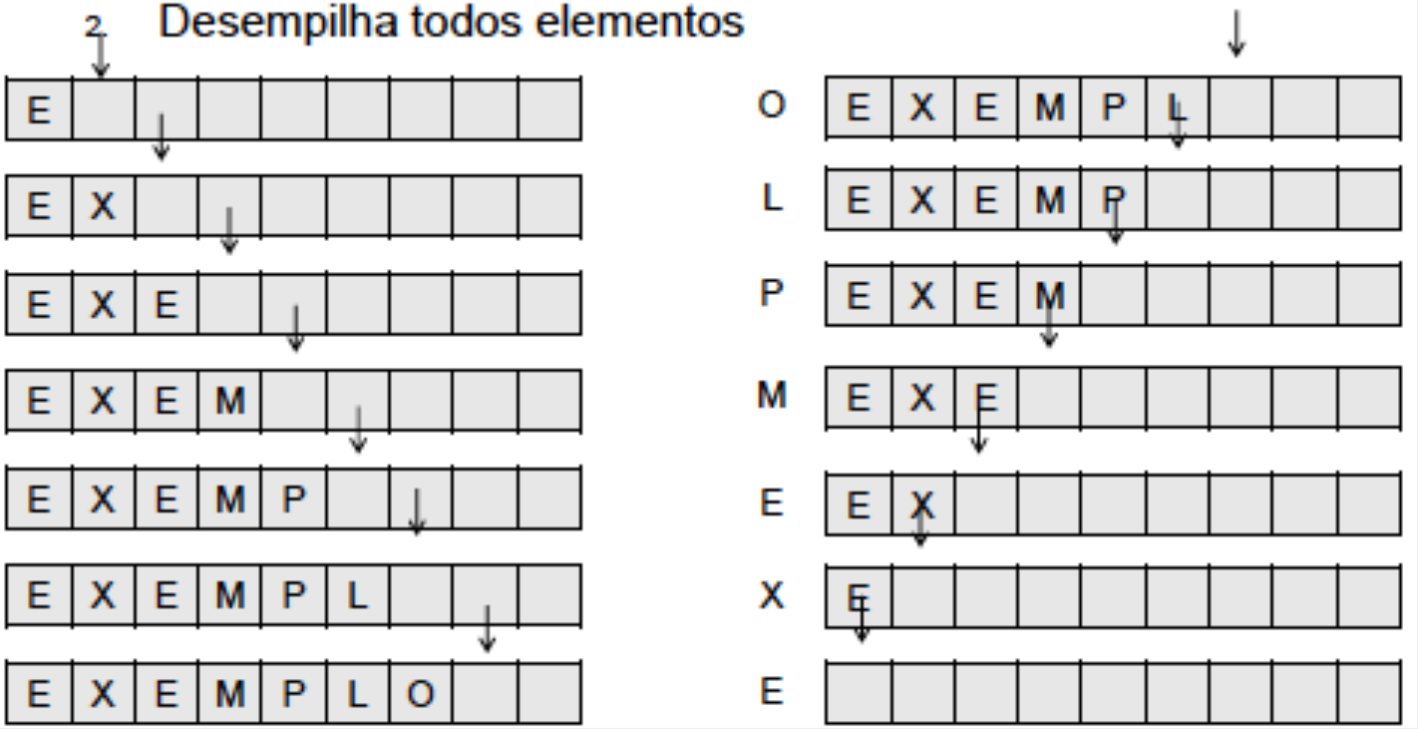


Operações sobre Pilhas Usando Apontadores

```
int Tamanho(TipoPilha *Pilha)
{
  return (Pilha->Tamanho);
}  /* Tamanho */
```

Exemplos do Uso de Pilhas

- Inverter a string "Exemplo" usando uma pilha.
  1. Empilha cada caractere em uma pilha vazia
  2. Desempilha todos elementos



Exercício de Fixação

- Implemente uma pilha usando arranjos
- Implemente uma função para imprimir os elementos da pilha seguindo a ordem do LIFO
- Escreva um algoritmo que use uma pilha para inverter a ordem das letras de cada palavra de uma string ASCII, preservando a ordem das palavras. Por exemplo, para a string ESTE EXERCICIO E MUITO FACIL o resultado deve ser ETSE OICICREXE E OTIUM LICAF.
- Faça um programa em C++ para ler um número inteiro maior que zero, converter este número de decimal para binário, usando pilha e apresentar na tela, o resultado da conversão.