PCC104 - Projeto e Análise de Algoritmos

Marco Antonio M. Carvalho

(baseado nas notas de aula do prof. Túlio A. M. Toffolo)
Departamento de Computação
Instituto de Ciências Exatas e Biológicas
Universidade Federal de Ouro Preto

Conteúdo

Fonte

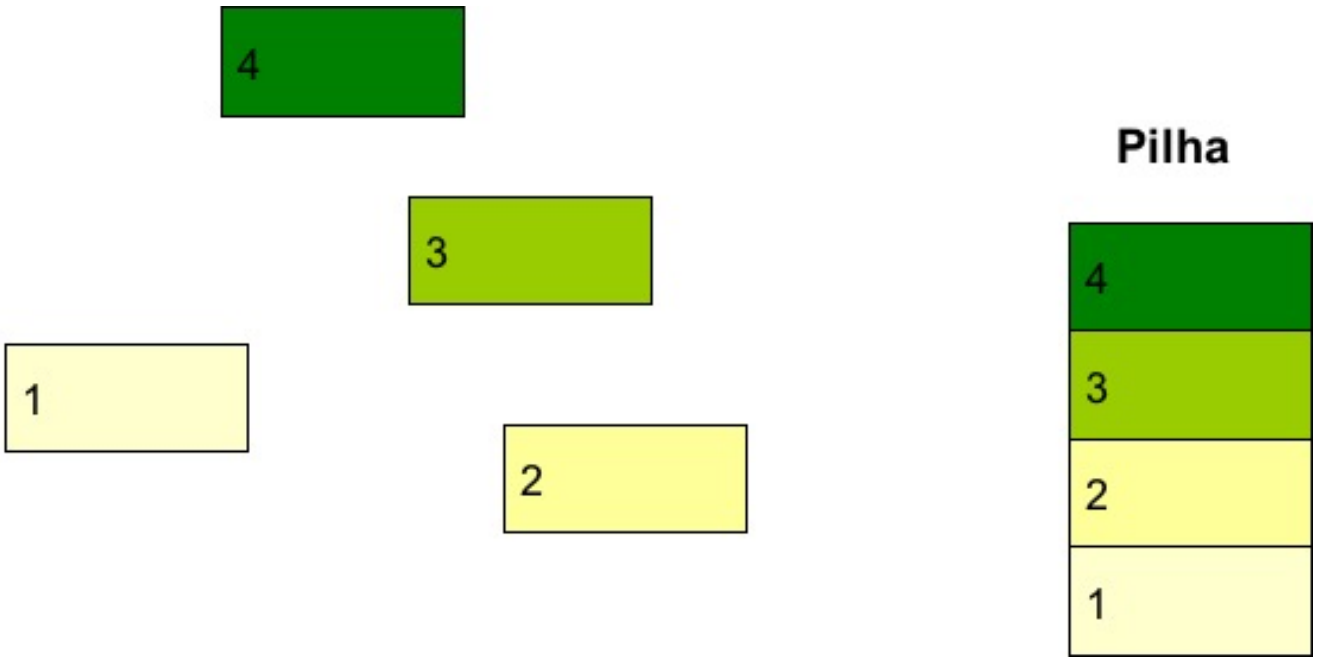
Este material é baseado nos livros

- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.
- S. Halim. *Competitive Programming*. 3rd Edition, 2013.
- Ian Parberry and William Gasarch. *Problems on Algorithms*. Second Edition, 2002.
- Ian Parberry *Lecture Notes on Algorithm Analysis and Complexity Theory*. Fourth Edition, 2001.

Licença

Este material está licenciado sob a Creative Commons BY-NC-SA 4.0. Isto significa que o material pode ser compartilhado e adaptado, desde que seja atribuído o devido crédito, que o material não seja utilizado de forma comercial e que o material resultante seja distribuído de acordo com a mesma licença.

Pilhas



Pilhas

Descrição

Pilhas são um tipo abstrato de dados com a característica de que o último elemento a ser inserido é o primeiro a ser removido (política LIFO – *Last in First Out*).

Considera-se uma analogia com pilha de elementos, como pratos, livros, etc.

Os usos de pilhas incluem a chamada de subprogramas, avaliação de expressões aritméticas, etc.

Pilhas

Operações

Pilha_Inicia(Pilha) Inicializa uma pilha vazia.

Pilha_Push(Pilha, x) Empilha (*push*) o item x no topo da pilha.

Pilha_Pop(Pilha, x) Desempilha (*pop*) o item x do topo da pilha.

Pilha_Tamanho(Pilha) Esta função retorna o número de itens da pilha.

Pilha_EhVazia(Pilha) Retorna 1 se a pilha está vazia; caso contrário, retorna 0.

Forma de Implementação

Existem várias opções de estruturas de dados que podem ser usadas para representar pilhas. As duas representações mais utilizadas são:

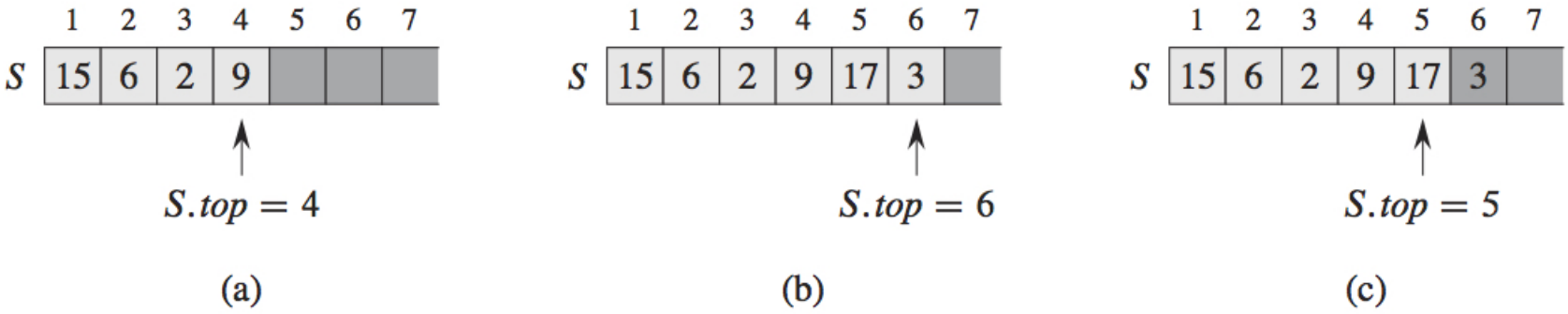
- Por meio de arranjos.
- Por meio de ponteiros.

Independente da forma de implementação, uma pilha é uma lista com restrições quanto às formas de inserção e remoção, o que permite a reusabilidade de código.

Implementação por Arranjos

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

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



(a) Uma pilha com quatro elementos. (b) Após empilhar dois elementos. (c) Após desempilhar um elemento.

Implementação por Ponteiros

Criar um campo **tamanho** evita a contagem do número de itens na função tamanho.

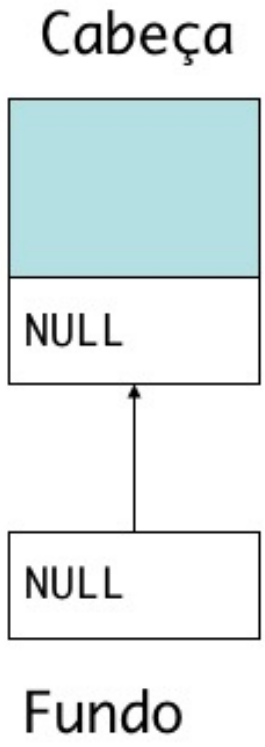
Cada célula de uma pilha contém um item da pilha e um apontador para outra célula.

A estrutura contém um apontador para o topo da pilha (célula cabeça) e um apontador para o fundo da pilha.

Ambos funcionam como início e fim de uma lista encadeada, respectivamente.

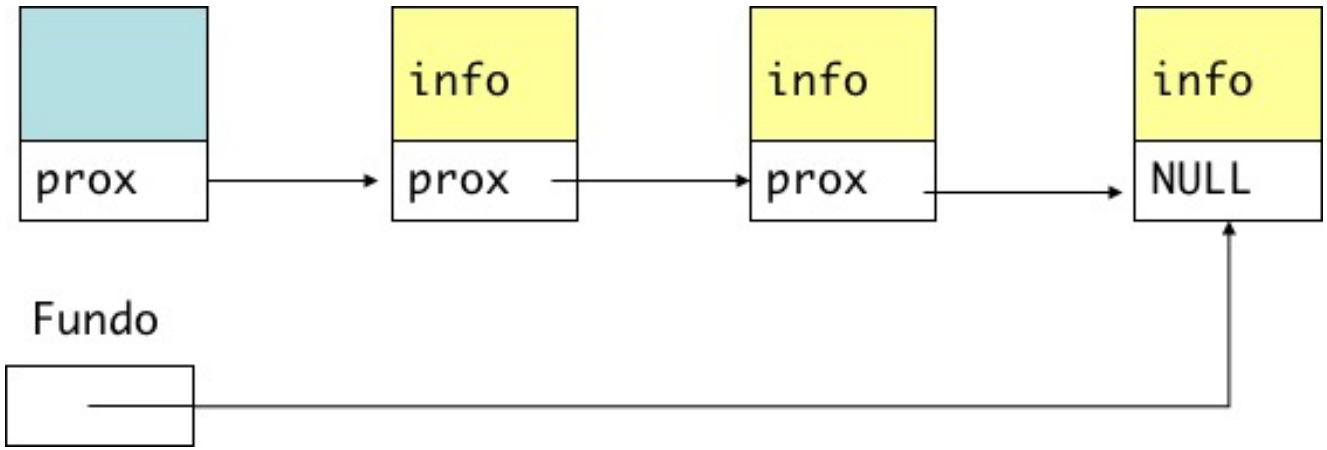
Implementação por Ponteiros Usando Célula Cabeça

Para criarmos uma pilha vazia, podemos fazer com que o fundo e o topo sejam o mesmo elemento.

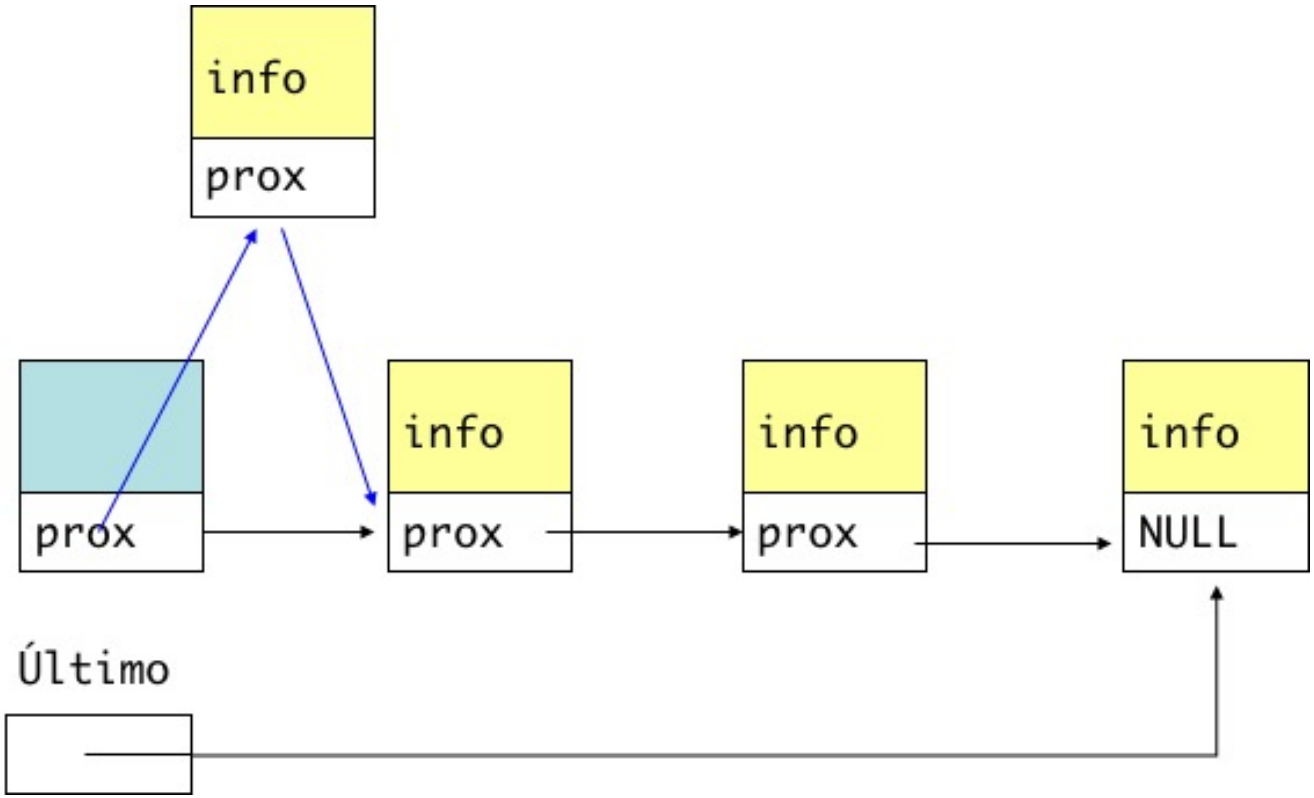


Inserção de Elementos

De acordo com a política LIFO, há apenas uma opção de posição onde podemos inserir elementos: o topo da pilha (ou seja, primeira posição)

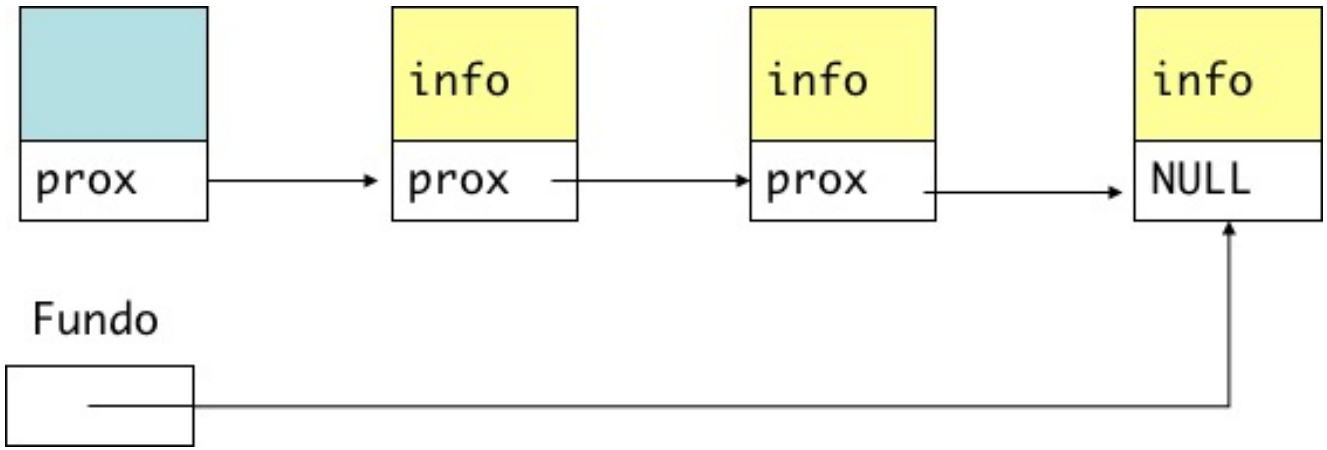


Pilhas

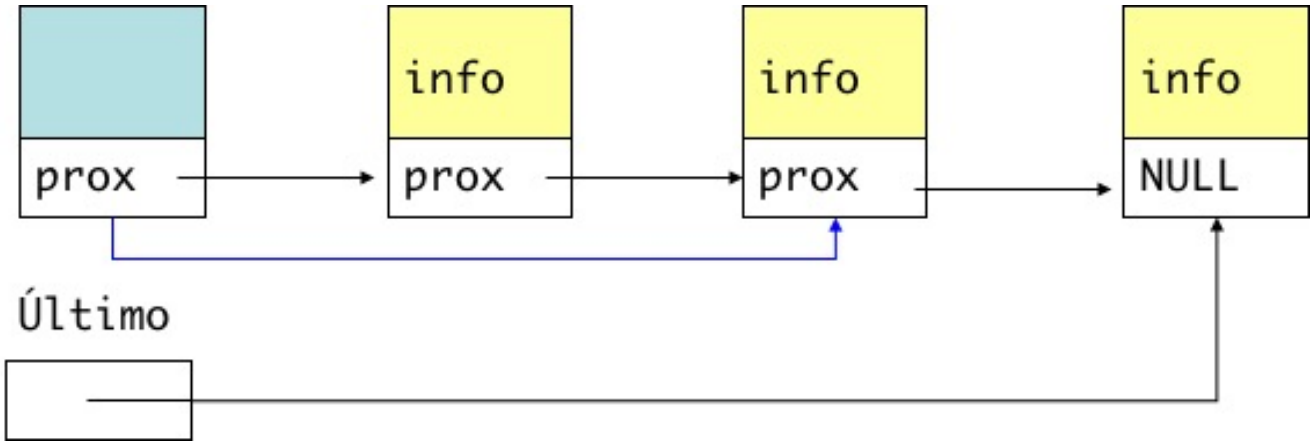


Remoção de Elementos

De acordo com a política LIFO, há apenas uma opção de posição onde podemos remover elementos: o topo da pilha (ou seja, primeira posição).



Pilhas



Complexidade

A complexidade de todas as operações é mantida da implementação de Lista:

- Empilhar: $\Theta(1)$;
- Desempilhar: $\Theta(1)$.

Exemplo

Vamos projetar um Editor de Texto que aceite os comandos:

- Cancela caractere;
- Cancela linha;
- Imprime linha.

O Editor de Texto 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 Editor de Texto deverá utilizar uma pilha para organizar os dados.

Exemplo

- # : cancela o 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.

Instância Exemplo

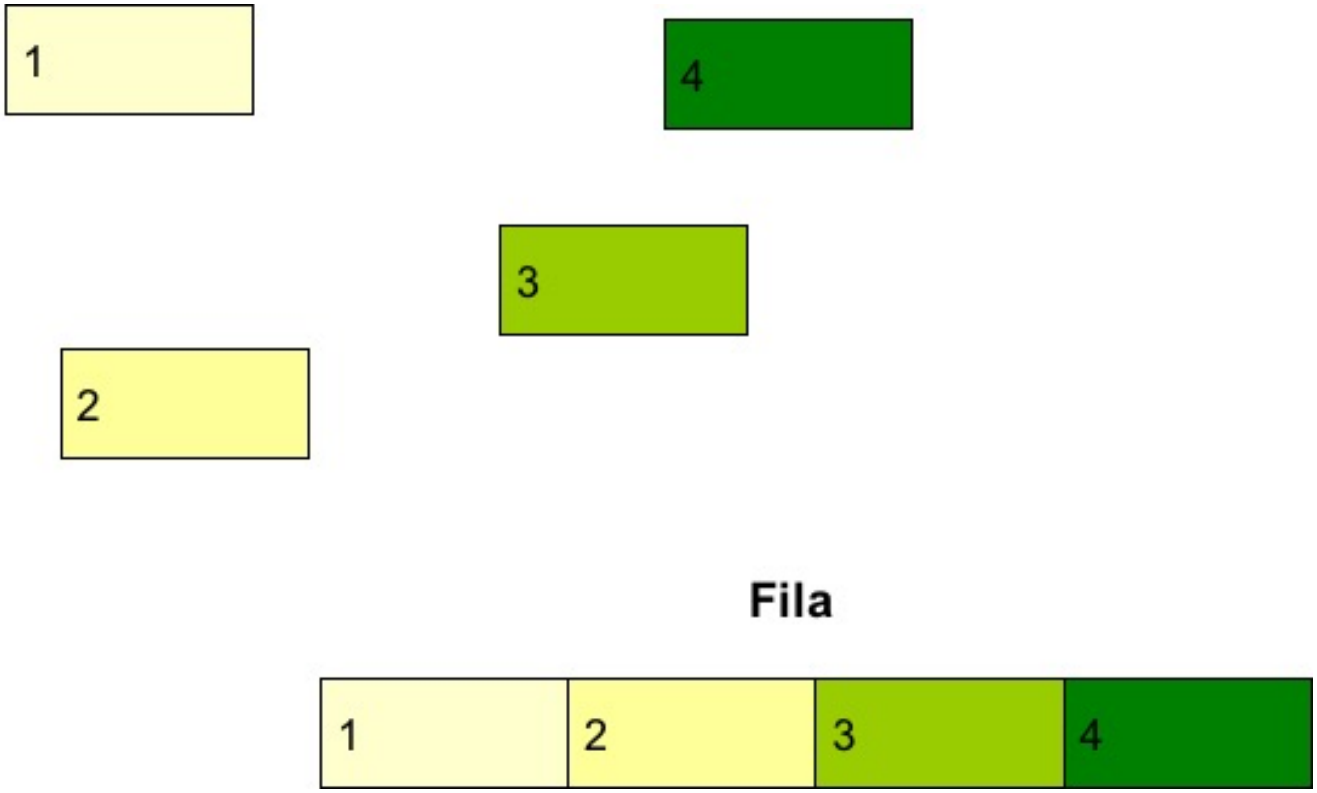
```
 Este et# um teste para o ET, o extraterrestre em
C.*Acabamos de testar a capacidade de o ET saltar de linha,
utilizando seus poderes extras (cuidado, pois agora vamos
estourar a capacidade máxima da linha de impressão, que é de
70 caracteres.)*O k#cut#rso dh#e Estruturas de Dados et#
h#um cuu#rsh#o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

```
int main(int argc, char* argv[]){
    TPilha Pilha;
    TItem x;
    TPilha_Inicia(&Pilha);
    x.Chave = getchar();
    while (x.Chave != MarcaEof) {
        if (x.Chave == CancelaCaractere)
            if (!TPilha_EhVazia(&Pilha))
                TPilha_Pop(&Pilha, &x);
        else if (x.Chave == CancelaLinha)
                TPilha_Inicia(&Pilha);
        else if (x.Chave == SaltaLinha)
                TPilha_Imprime(&Pilha);
        else if (TPilha_Tamanho(Pilha) == MaxTam)
                TPilha_Imprime(&Pilha);
        else
                TPilha_Push(&Pilha, x);
        x.Chave = getchar();
    }
    if (!TPilha_EhVazia(&Pilha))
                TPilha_Imprime(&Pilha);
    return 0;
}
```

Exemplo

```
void PImprime(TipoPilha* pPilha)
{
    TPilha Pilhaux;
    TItem x;
    TPilha_Inicia(&Pilhaux);
    while (!TPilha_EhVazia(pPilha)) {
        TPilha_Pop(pPilha, &x);
        TPilha_Push(&Pilhaux, x);
    }
    while (!TPilha_EhVazia(&Pilhaux)) {
        TPilha_Pop(&Pilhaux, &x);
        putchar(x.Chave);
    }
    putchar('\n');
}
```

Filas



Descrição

Filas são um tipo abstrato de dados com a característica de que o primeiro elemento a ser inserido é o primeiro a ser removido (política FIFO – *First in First Out*).

Considera-se uma analogia com filas de elementos, como pessoas, processos, etc.

Os usos de filas incluem filas de impressão e filas de processamento em sistemas operacionais, entre outros.

Operações

Fila_Inicia(Fila) Inicia uma fila vazia.

Fila_Enfileira(Fila, x) Insere (*enqueue*) o item x no final da fila.

Fila_Desenfileira(Fila, x) Remove (*dequeue*) o item x no início da fila.

Fila_Tamanho(Fila) Esta função retorna o número de itens da fila.

Fila_EhVazia(Fila) Retorna 1 se a fila está vazia; caso contrário, retorna 0.

Forma de Implementação

Existem várias opções de estruturas de dados que podem ser usadas para representar filas. As duas representações mais utilizadas são:

- Por meio de arranjos.
- Por meio de ponteiros.

Independente da forma de implementação, uma fila é uma lista com restrições quanto às formas de inserção e remoção, o que permite a reusabilidade de código.

Implementação por Arranjos

Os itens são armazenados em posições contíguas de memória.

A operação **Enfileira** faz a parte de trás da fila expandir-se.

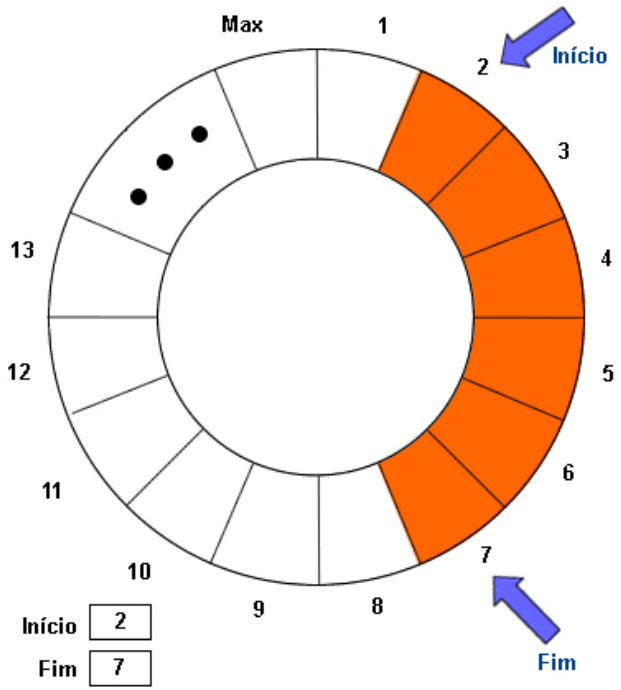
A operação **Desenfileira** faz a parte da frente da fila contrair-se.

A fila tende a caminhar pela memória do computador, ocupando espaço na parte de trás e descartando espaço na parte da frente.

Implementação por Arranjos

Ao longo de inserções e retiradas, a fila vai ao encontro do limite do espaço da memória alocado para ela.

Uma solução é imaginar o arranjo como um círculo, em que a primeira posição segue a última.



Implementação por Arranjos

A fila se encontra em posições contíguas de memória, em alguma posição do círculo, delimitada pelos apontadores **Início** e **Fim**.

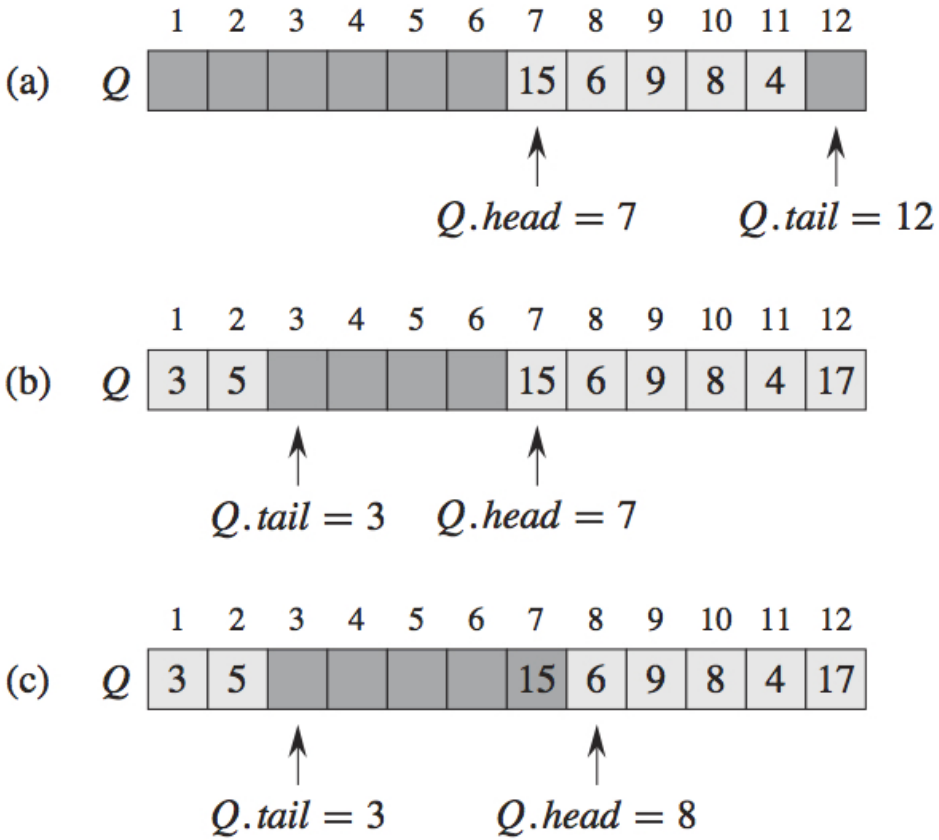
Início indica a posição do primeiro elemento.

Fim a primeira posição vazia (posição após o último elemento).

Para enfileirar, basta mover o apontador **Fim** uma posição no sentido horário.

Para desenfileirar, basta mover o apontador **Início** uma posição no sentido horário.

Filas



(a) Fila com cinco elementos. (b) Após o enfileiramento de três elementos. (c) Após desenfileirar um elemento.

Implementação por Ponteiros

Há uma célula cabeça para facilitar a implementação das operações **Enfileira** e **Desenfileira** quando a fila está vazia.

Quando a fila está vazia, os apontadores **Início** e **Fim** apontam para a célula cabeça.

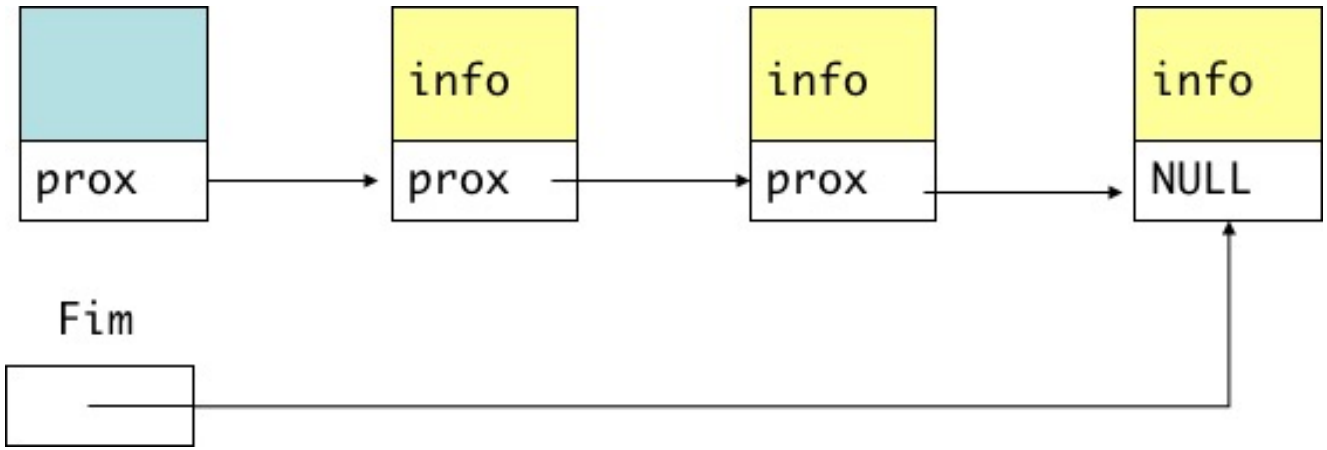
Para enfileirar um novo item, basta criar uma célula nova, ligá-la após a célula que contém x_n e colocar nela o novo item.

Para desenfileirar o item x_1, basta desligar a célula após a cabeça da lista.

Implementação por Ponteiros

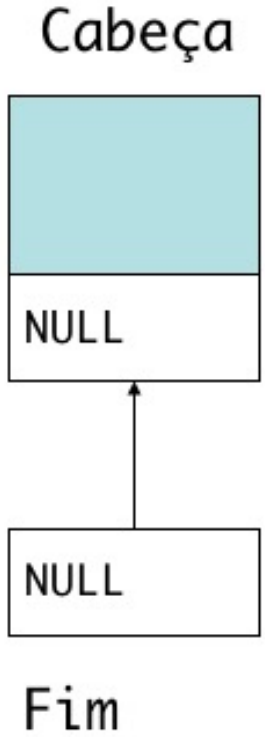
A fila é implementada por meio de células, tal que cada célula contém um item da fila e um apontador para a próxima célula.

A estrutura contém um apontador para a **frente** da fila (célula cabeça) e um apontador para a parte de **trás** da fila (fim).



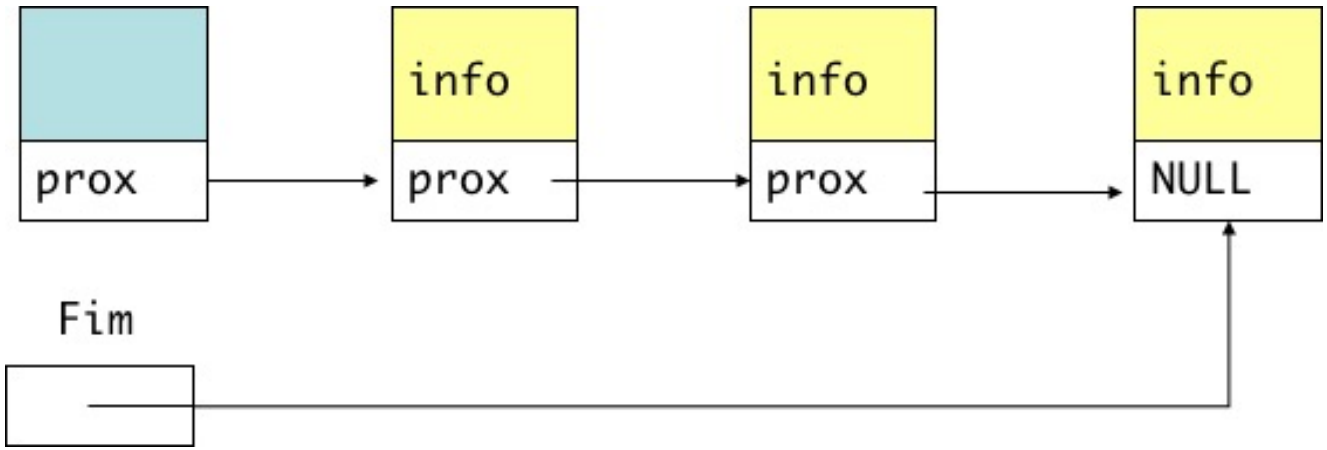
Implementação por Arranjos Usando Célula Cabeça

Para criarmos uma fila vazia, podemos fazer com que o início e o fim sejam o mesmo elemento.

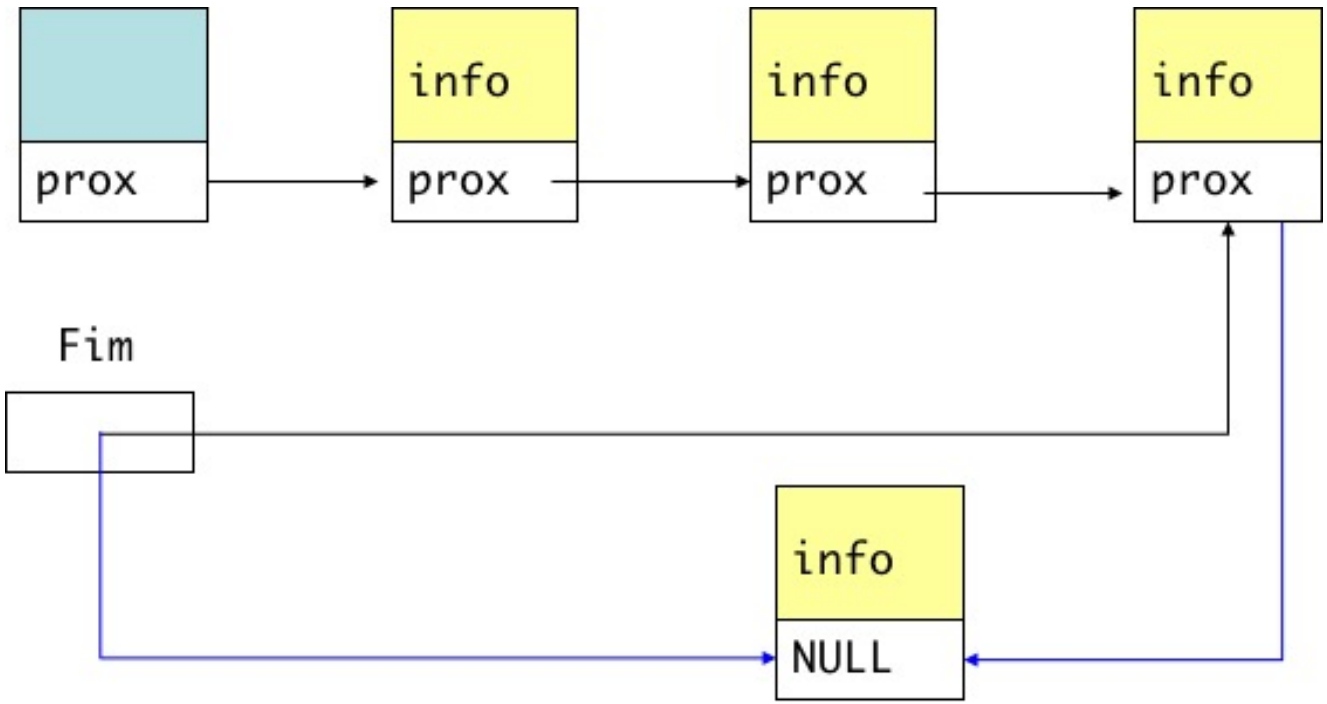


Inserção de Elementos

De acordo com a política FIFO, há apenas uma opção de posição onde podemos inserir elementos: o fim da fila (ou seja, a última posição) .

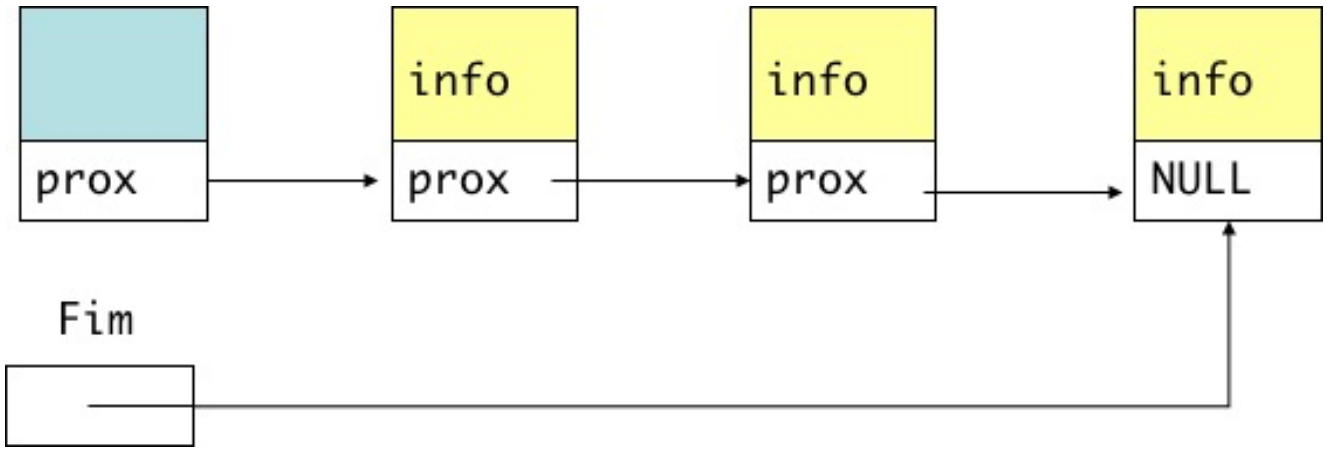


Filas

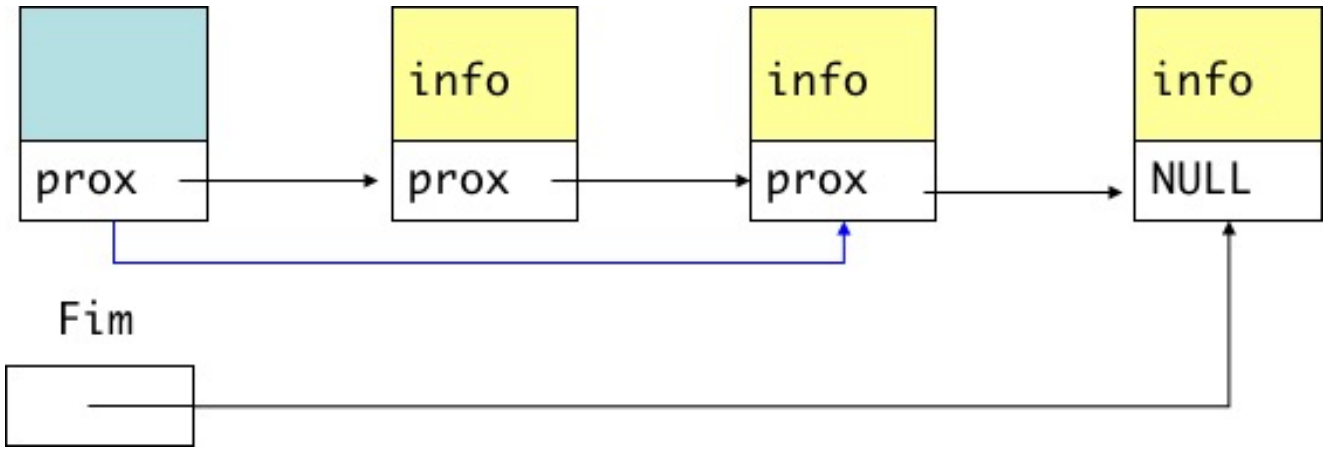


Remoção de Elementos

De acordo com a política FIFO, há apenas uma opção de posição onde podemos remover elementos: o início da fila (ou seja, primeira posição).



Filas



Complexidade

A complexidade de todas as operações é mantida da implementação de Lista:

- Enfileirar: $\Theta(1)$.
- Desenfileirar: $\Theta(1)$.

Exercício 1

Seja **TLista** o tipo abstrato de dados Lista que implementa as funções abaixo.

Supondo que você tem conhecimento de que **TLista** é uma lista encadeada, implemente a tipo abstrato de dados **TFila** utilizando um **TLista**

```
void TLista_Inicia(TLista *pLista);
int TLista_EhVazia(TLista *pLista);
int TLista_InsereFinal(TLista *pLista, TItem x);
int TLista_InserePrimeiro(TLista *pLista, TItem x);
int TLista_RetiraPrimeiro(TLista *pLista, TItem *pX);
int TLista_RetiraFinal(TLista *pLista, TItem *pX);
int TLista_Tamanho(TLista *pLista);
```

Exercício 2

Seja **TLista** o tipo abstrato de dados Lista que implementa as funções abaixo.

Supondo que você tem conhecimento de que **TLista** é uma lista encadeada, implemente a tipo abstrato de dados **TPilha** utilizando um **TLista**

```
void TLista_Inicia(TLista *pLista);
int TLista_EhVazia(TLista *pLista);
int TLista_InsereFinal(TLista *pLista, TItem x);
int TLista_InserePrimeiro(TLista *pLista, TItem x);
int TLista_RetiraPrimeiro(TLista *pLista, TItem *pX);
int TLista_RetiraFinal(TLista *pLista, TItem *pX);
int TLista_Tamanho(TLista *pLista);
```

Dúvidas?

Exercício 3

É recomendável implementar uma pilha por ponteiros (encadeamento simples) tal que as inserções e remoções sejam feitas sempre no final da estrutura? Justifique.

Exercício 4

Mostre como implementar uma pilha usando duas filas. Analise o tempo de execução das operações da implementação sugerida.

Dúvidas?