Estruturas de Dados - I

Estudo do comportamento e operação da estrutura Pilha/Multipilha encadeada na linguagem C

Alunos: Luciano Kohler e Pedro Stupp
Professor: Gilmário Barbosa dos Santos

Joinville - 2025

SUMÁRIO

1 INTRODUÇÃO

Uma **pilha encadeada** é um tipo de dado estruturado no ramo do armazenamento de informações que, similarmente à uma lista encadeada, armazena valores que se conectam com o seu próximo valor via um **apontador** (comumente chamado de **prox**), porém, ao contrário de uma lista, uma pilha encadeada implementa o comportamento de LIFO (Last In, First Out), que como o nome indica, o valor mais recente a ser inserido na pilha, também será o primeiro valor a ser encontrado/removido da pilha.

Neste estudo, veremos a implementação de uma ramificação da pilha (denominada “multipilha”), e um uso prático de pilha para a verificação sintática de arquivos .html

2 OBJETIVOS

GERAL

- Realizar uma análise compreensiva do funcionamento de uma pilha encadeada e seus usos práticos.

ESPECÍFICOS

- Implementar via codificação em C, uma multipilha funcional e seus diferentes métodos.
- Sintetizar um uso prático de uma estrutura de pilhas por meio da criação de um programa em C que, dado um arquivo de formato .html, leia suas *tags* e informe se a sua estrutura está sintaticamente correta.

3 METODOLOGIA

3.1 Recursos para o experimento

Utilização de um notebook Dell Inspiron 15:

- Memória RAM: 16GB
- Processador: Intel Core i5-1235U 10 núcleos
- Tipo de unidade de armazenamento: SSD M.1 512GB

3.2 Estudo de multipilhas

Para a implementação da multipilha, foi criado um ponteiro *descritor*, que possui atributos responsáveis pela descrição e acesso de itens da multipilha com os seguintes valores:

- **TopoDireita:** Armazena o primeiro elemento da pilha à direita
- **TopoEsquerda:** armazena o primeiro elemento da pilha à esquerda
- **tamVet:** Armazena o tamanho que cada uma das duas pilhas possui

Após isso, foi implementado os seguintes métodos para a manipulação da multipilha:

- **criaPilha()**: Cria um novo descritor pronto para armazenar valores da pilha
- **reiniciaPilha()**: Dá *free()* todos os valores da pilha de ambos os seus lados
- **destruirPilha()**: Reinicia a pilha E dá *free()* no ponteiro descritor
- **empilha()**: Insere um novo nodo à direita ou à esquerda da pilha
- **desempilha()**: Remove o elemento mais acima da pilha direita ou esquerda
- **buscaNoTopo()**: Retorna o elemento mais no topo da fila direita ou esquerda
- **checaSeCheia()**: Indica se a pilha está cheia em um dos lados
- **pilhaVazia()**: Indica se a pilha está vazia em um dos lados
- **tamanhoPilha()**: Retorna a quantidade de valores de um lado da pilha

3.3 Avaliador de sintaxe de HTML

Para a implementação desse sistema, foi feito um sistema muito similar ao sistema anterior da multipilha, porém desta vez, utilizando uma pilha encadeada simples, pois o uso da multipilha neste cenário seria redundante, pois um dos lados da pilha estaria sempre vazio e sem utilidade, então, foi escolhido a implementação de uma pilha de um lado, pois assim, é possível armazenar as aberturas de tags nela, e a remoção dos elementos (fechamento da tag) tudo na mesma pilha.

O descritor é similar ao antigo, porém agora somente com um topo, e as funções implementadas são idênticas ao da multipilha, porém, sem a escolha de realizar a operação do lado direito ou esquerdo da pilha (já que a pilha só tem um lado).

Adicionalmente, foi implementado a função **lerHTML()**, que recebe um caminho para um arquivo .html, abre-o, separa-o em tags (inicialmente recebe todas as linhas, porém vai removendo linhas em branco ou conteúdos entre as tags), e segue três caminhos dependendo da tag recebida:

- Se a tag for de abertura (exemplo: <head>), a função simplesmente irá inserir a tag na pilha
- Se a tag for de fechamento(exemplo: <\head>), a função irá comparar essa tag com a tag no topo da fila, se elas forem iguais, isso indica que houve o fechamento correto das tags, caso contrário, o código retorna um erro, indicando que o código html não está estruturado corretamente
- Se a tag for “solitária” (não abre nem fecha, como por exemplo: <!DOCTYPE html>, o código ignora a tag, afinal, essa tag sempre estará correta por não precisar se fechar

  Ao final do código, é retornado se o arquivo está sintaticamente correto, ou não, informando qual tag precisava ser fechada para o funcionamento do código.

4 CONCLUSÃO

Após a aplicação prática de ambos os cenários requisitados pelo professor, é possível notas cenários práticos em que a pilha é de muita usabilidade, e tirando o cenário visto na prática, também temos diversos outros tipos de usos cotidianos em que a fila prova seu valor, como por exemplo:

- Salvamento de modificações em documento para implementar a funcionalidade de desfazer (ctrl + z) e refazer (ctrl + y)
- Cálculo de expressões matemáticas com parênteses
- Checagem de sintaxe de outras estruturas similares à HTML, como por exemplo, o XML

5 REFERÊNCIAS BIBLIOGRÁFICAS

What is Stack Data Structure? A Complete Tutorial. *GeeksForGeeks,* 06 Mar. 2025, Disponível em: <https://www.geeksforgeeks.org/introduction-to-stack-data-structure-and-algorithm-tutorials/>. Acesso em: 03 Jun. 2025