ESTRUTURA DE DADOS I - Relatório Dos Exercícios 1 E 2 Da Tarefa 1.A

ANA ELISA GHANEM ZANON E LUCAS ANAND MAZOTTI FERRETTI

EXERCÍCIO 1:

Em todos os casos verificamos se já existe uma pilha.

- No caso 1 o usuário escolhe em qual pilha ele quer adicionar um novo elemento e então ele fornece o elemento. Após isso, o algoritmo realiza o empilhamento do elemento.
- O caso 2 serve para desempilhar um objeto da pilha é solicitado escolher qual das duas pilha quer retirar o objeto,caso o objeto seja diferente de 0 é desempilhado o objeto, caso contrário é mostrado uma mensagem dizendo “pilha vazia”.
- O caso 3 busca o elemento no topo da pilha, solicitamos em qual das duas pilhas queremos ver o topo e o programa informa o elemento no topo se a pilha não estiver vazia, caso esteja é mostrado uma mensagem dizendo “pilha vazia”.
- O caso 4 informa o tamanho da pilha, o usuário digita qual pilha quer ver o tamanho e é informado o tamanho da pilha
- O caso 5 reinicia a pilha, o usuário informa qual pilha deseja reiniciar e o programa volta ao topo da pilha para o início(caso seja a primeira) ou final(caso seja a segunda)
- O caso 6 apaga as pilhas, o programa apaga o ponteiro da multipilha.
- O caso 7 cria uma nova pilha, o programa verifica se já tem uma pilha existente, caso tenha o programa destrói a pilha existente e cria uma nova com o tamanho informado pelo usuário.

Caso o usuário não escolhe nenhuma das 7 opções acima retorna uma mensagem de opção inválida

Para as funções:

`criaPilha`: Quando criamos uma multipilha, instanciamos o descritor e inserimos o tamanho do vetor, definimos o topo1 para -1 e o topo2 como o tamanho do vetor e por fim criamos um vetor com o tamanho passado. Após isso retornamos um ponteiro para a multipilha.

`destroiPilhas`: Para destruir a pilha damos free no vetor e definimos como NULL o endereço de memória da multipilha

`reiniciaPilha`: Para reiniciar a pilha, o programa volta ao topo da lista escolhida para o início dela, caso seja a pilha 1, ou para o final dela, caso seja a pilha2.

`empilha`: Após ser escolhido qual pilha deve ser empilhado, verificamos se a pilha está vazia e caso esteja adicionamos o elemento passado pelo usuário e aumentamos ou diminuímos o topo, dependendo de qual pilha é.

`desempilha:` Verificamos se a pilha está vazia, caso contrário retiramos o elemento do topo da pilha e adicionamos +1 ou -1 no índice do topo da pilha, por fim retornamos o número que foi retirado.

`testeCheia:` Verificamos se o topo +1 da pilha1 é igual ao topo2 ou se o topo2-1 é igual ao topo1

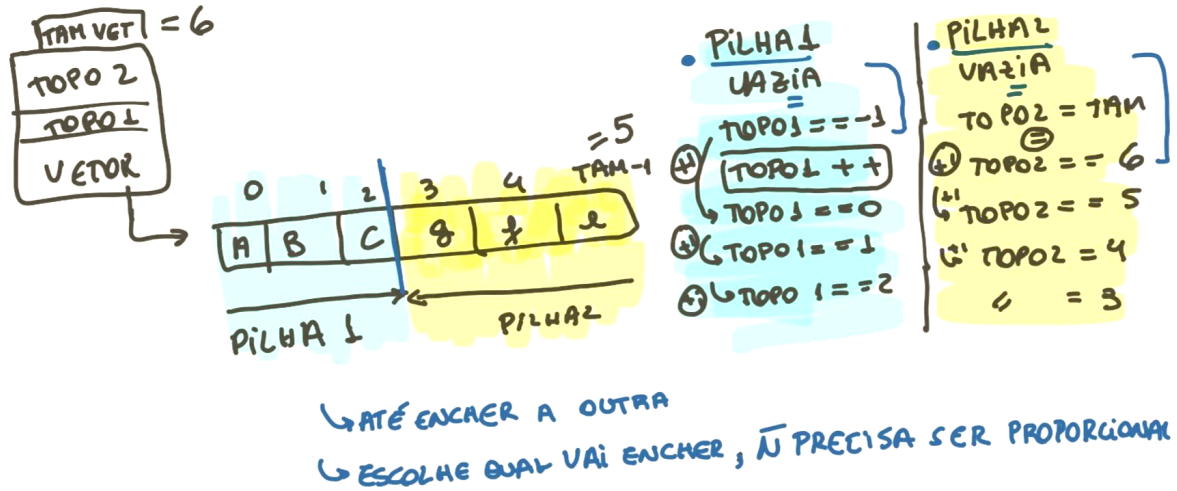
`testeVazia:` Verificamos se o topo das pilhas é -1(pilha1) ou o tamanho do vetor(pilha2)

`buscaTopo:` Verificamos se a pilha está vazia, caso não esteja retornamos o elemento que está no topo1 ou topo2 do vetor.

`tamanhoPilha:` para a pilha1 apenas somamos 1 no topo e retornamos, mas na pilha2 fazemos a diferença entre o tamanho do vetor e o topo da segunda pilha.

Foi compilado em um ambiente linux(Arch) usando o VsCode e o compilador foi o gcc com o comando **`gcc main.c pilha.c pilha.h`**, depois foi executado o arquivo **`a.out`**, as bibliotecas usadas foram a **`<stdlib.h>`** e **`<stdio.h>`**

TESTE DE MESA TAREFA 1:



EXERCÍCIO 2:

Conforme o enunciado, para esse exercício foi utilizado um arquivo de pilha disponível no Moodle da disciplina. Arquivo: **`pilhaSE.c`**

Justificativa da escolha da pilha encadeada: Escolhemos essa implementação da pilha encadeada, pois não teremos que lidar com o limite de tamanho que ocorre quando utilizamos um vetor.

Inicialmente é solicitado ao usuário o caminho do arquivo em HTML então, é feita a leitura do arquivo. Em caso de não encontrar o arquivo uma mensagem de “Arquivo não encontrado” é exposta na tela. Então, utilizando o arquivo é criada uma Pilha em que é definido que o programa será rodado até encontrar a tag **`</html>`**. A leitura das tags de

HTML é feita de forma que sejam ignoradas as seguintes tags (que não precisam/não tem fechamento): **`meta, br, img, input, frame`** e **`!DOCTYPE`**. E no caso das tags de comentário **`<!-- -->`** não será tratada sendo ignorada na pilha e tudo que estiver dentro dela (mesmo que não haja fechamento). As tags de abertura serão lidas e colocadas na pilha caso tenham início "**`<`**" e não possuam "**`/`**" dentro da tag.

Na pilha, o desempilhamento da tag do topo só será possível quando encontrar a tag de fechamento com início "**`<`**" e que possuam "**`/`**" dentro da tag. Por meio do **`strcmp`** é feita a verificação se a palavra encontrada dentro da tag de início é a mesma palavra da tag de fechamento, o que permite o desempilhamento da tag de início da pilha com a chamada da função **`desempilha`**. Também há o caso de haver um empilhamento sem ocorrer o anterior fechamento de uma tag do topo, então o programa apresenta uma mensagem de **`"ERRO: esperado (tag esperada) e recebido (tag incorreta)"`**.

Para a compilação utilizamos o gcc em um ambiente linux(Arch), o comando utilizado foi:
gcc main.c pilhaSE.c arq.h e **a.out a.html** para poder ler o arquivo. As bibliotecas usadas foram **`<stdio.h>`**, **`<stdlib.h>`** e **`<string.h>`** além das bibliotecas da pilha simplesmente encadeada "**`arq.h`**" que pegamos do moodle.

TESTE DE MESA TAREFA 2:

- Arquivo HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
        <title> Document </title>
</head>
<body>
        <br>
</body>
</html>
```

Repetição = 0;
atual = "<! DOCTYPE";
- false no if de exceção;
pilha = vazia;

Repetição = 1;
atual = "html>";
- não entra em nenhum if e sai;

Repetição = 2;
atual = "<html"
- Como atual[0] == '<' e atual[1] != '/';
pilha = ["html"];

Repetição = 3;
atual = "lang="en">"
- não é tag html;

Repetição = 4;
atual = "<head>";
- é uma abertura de tag
pilha = ["head"] → ["html"]

Repetição = 5
atual = "<title>",
- é uma abertura de tag
pilha = ["title"] → ["head"] → ["html"]

Repetição = 6
atual = "Document"
- não é tag html

Repetição = 7
atual = "</title>"
- é fechamento de tag
Pilha = ["head"] → ["html"]

Repetição = 8
atual = "</head>"
- é fechamento de tag
Pilha = ["html"]

Repetição = 9
atual = "<body>"
- é abertura de tag
Pilha = ["body"] → ["html"]

Repetição = 10
atual = "
"
- Cai na exceção

Repetição = 11
atual = "</body>"
- fechamento de tag
Pilha = ["html"]

Repetição = 12
atual = "</html>"
- fechamento de tag
Pilha = vazia

- encerra sem erros!

- Arquivo HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
        <title> Document </title>
</head>
<body>
        <br>
</html>
```

a mesma sequência do anterior, porém quando a repetição chegar em 11 muda para esta parte