UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

MATHEUS GOMES ARANTE DE SOUZA
MATHEUS SALOMÃO

RELATÓRIO REFERENTE AO 1º TRABALHO PRÁTICO DA DISCIPLINA ESTRUTURA DE DADOS I

VITÓRIA
2018

MATHEUS GOMES ARANTE DE SOUZA
MATHEUS SALOMÃO

RELATÓRIO REFERENTE AO 1º TRABALHO PRÁTICO DA DISCIPLINA ESTRUTURA DE DADOS I

Trabalho realizado para colocar em prática a manipulação do uso de tipos abstratos de dados e estruturas do tipo lista na linguagem de programação C.

VITÓRIA
2018

SUMÁRIO

INTRODUÇÃO

Neste relatório abordaremos o desenvolvimento do trabalho e as soluções que foram pensadas para definir as funções propostas.

Com o objetivo de implementar a enciclopédia colaborativa denominada WikED o programa foi dividido em tipos abstratos de dados, dos quais contém estruturas de dados e conjuntos de funções. Cada TAD foi criado com objetivo de modularizar o programa assim como torná-lo mais legível e organizado.

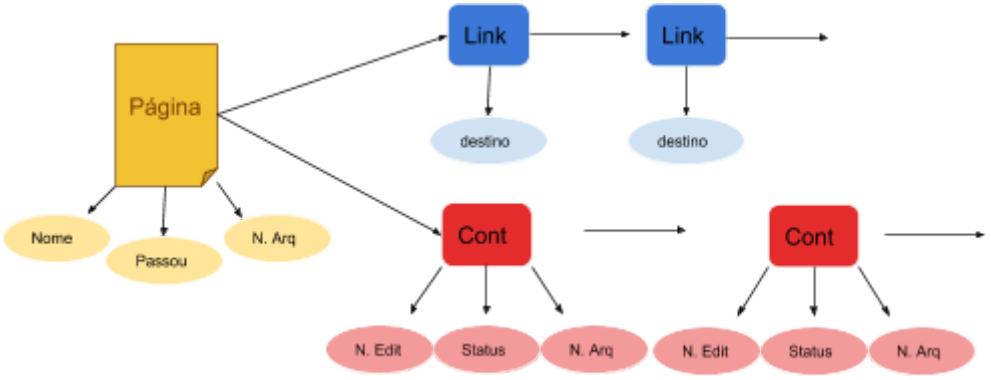
Este relatório foi dividido a partir dos TADs criados, portanto, cada um possui uma seção especial explicando suas estruturas e funções. As estruturas criadas possuem diagramas ilustrativos no início da seção correspondente ao TAD em que elas estão definidas.

Para que o código possa ser compreendido da melhor forma possível recomendamos que esse relatório seja lido junto ao código, visto que em determinadas funções, como a *caminho*, os comentários presentes no código complementam a explicação do relatório.

OS TIPOS ABSTRATOS DE DADOS

A WikED foi modularizada em seis TADs: páginas, editores, contribuições, links, print e arquivo de log, de tal forma que todos eles se relacionam.

PÁGINA



No TAD página criamos as structs que compõe a lista encadeada de páginas: *listapagina*, *celulapagina* e *pagina*.

pagina (tipo opaco)
Contém campos:
- nome;
- arquivo;
- lista de contribuições;
- lista de links;
- passou.

celulapagina (tipo opaco)
Contém campos:
- *pagina;*
- ponteiro para a próxima *celulapagina.*

listapagina (tipo opaco)
Contém campos:
- Sentinela para a primeiro e a última célula da lista.

A função *inicializarPagina*, recebe o nome da página e o nome do arquivo. Dessa forma, estamos alocando a memória necessária e preenchendo seus campos, além de inicializar a lista de contribuições com a função *inicializarListaCont* e a lista de links *inicializarListaLink* desta página.

Para inserir uma página na WikED, utilizamos a função *inserePagina*, que recebe como parâmetros a lista de páginas, o nome dela e o nome do arquivo. Para implementá-la, utilizamos uma função auxiliar, chamada *inicializarPagina*. Após inicializar a página, finalmente inserimos ela na última posição da lista.

Para retirar uma página, utilizamos a função *retiraPagina*. Ela recebe como parâmetros a lista de páginas e o nome da página a ser retirada. Primeiramente, verificamos se a página em questão existe. Se ela existe, procuramos com uma variável auxiliar o ponteiro desta célula. Ao encontrá-lo, fazemos as devidas verificações sobre o posicionamento desta célula na lista e a liberamos com a função *liberaPagina*. Quando uma página é retirada, devemos verificar nas demais páginas quais possuem link com ela e remover este link. Portanto, utilizamos a função *retiraLinkAlternativa* com esse objetivo. Caso a página não exista, uma mensagem de erro é escrita no arquivo de log.

Para inserir um link de uma página de origem para uma página de destino, utilizamos a função *insereLinkPag*, que por sua vez chama a função *insereLink* para inserir na lista de links da página de origem, um link para a página de destino. Caso origem ou destino não existam, um erro é escrito no arquivo de log.

Para retirar um link de uma página de origem para uma página de destino, utilizamos a função *retiraLinkPag*, que por sua vez chama a função *retiraLink* para inserir na lista de links da página de origem, um link para a página de destino. Caso origem ou destino não existam, um erro é escrito no arquivo de log.

A função *retiraLinkAlternativa* é utilizada quando vamos retirar uma página e precisamos remover de todas as outras páginas os links que levam à esta página em questão. Para isso, percorremos por toda a lista de páginas e utilizamos a função *retiraLink* para nos auxiliar, passando a lista de links de cada página para ela, mas mantendo a mesma página de destino.

Para inserir uma contribuição em uma página utilizamos a função *insereContPag*. Inserimos a contribuição e suas demais informações na lista de contribuições desta página utilizando a função *insereContribuicao*. Caso esta página não seja encontrada, um erro é escrito no arquivo de log.

Para retirar uma contribuição de uma página, utilizamos a função *retiraContPag*. Buscamos na lista de contribuições dela a contribuição a ser retirada e tornamos o status desta igual a zero. Caso esta página não seja encontrada ou um editor diferente do que inseriu a contribuição esteja tentando removê-la, um erro é escrito no arquivo de log.

A função *retiraEditorListaPag* remove de todas as listas de contribuições das páginas da WikED as contribuições deste editor, utilizando a função auxiliar *retiraContPeloEditor*. Porém nesta função, ao invés de apenas tornar o status da contribuição igual a zero, tornando-a inválida, nós de fato retiramos a contribuição deste editor da lista de contribuições das páginas. E por fim, retiramos este editor da lista de editores com a função *retiraEditor*.

A função fim utiliza como auxiliar as funções *liberaListPag* e *liberaListEdit* para liberar toda a memória alocada.

A função *liberaListPag* recebe como parâmetro a lista de páginas e vai liberando cada célula da lista utilizando a função *liberaPagina*.

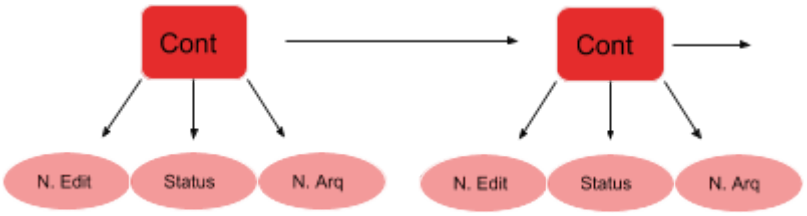
A função *liberaPagina* desaloca toda a memória alocada para uma célula de página, liberando o nome da página, nome do arquivo, sua lista de links e de contribuições.

A função *existePagina* recebe como parâmetros a lista de páginas e o nome de uma página. Percorremos esta lista com uma variável auxiliar para verificar se alguma das páginas é igual ao nome que foi dado como parâmetro. Caso o auxiliar não for nulo, significa que a página existe. Caso contrário, ela não existe. Utilizamos esta função como auxílio para diversas outras, quando precisamos verificar a existência de uma página.

A função *caminho* utiliza a função auxiliar *caminhoDireto* para descobrirmos se é possível chegar de uma página à outra utilizando os links entre elas. A *caminhoDireto* funciona da seguinte forma: ela recebe como parâmetros a lista de páginas, uma página origem e uma página destino. A partir daí, verificamos se na lista de links da página origem há um link para esta página destino. Se houver, retornamos 1. Caso contrário, chamamos a função recursivamente, de tal forma que dessa vez, o parâmetro página origem será ocupado pelo destino de uma célula de link. Este processo acaba quando percorrermos toda a lista de links da página de origem ou quando encontrarmos caminho.

As funções *retornaLstContPag*, *retornaLstLinkPag*, *retornaLstPagPrim*, *retornaPagProx*, *retornaNomePag* e *retornaNomeArq* foram criadas para que pudéssemos acessar um determinado campo das structs definidas no arquivo de página no arquivo de print.

CONTRIBUIÇÕES



No TAD contribuições criamos as structs que compõe a lista encadeada de contribuições: listacont, celulaCont e contribuicao.

contribuicao (tipo opaco)
Contém campos:
- nome do arquivo;
- nome do editor;
- status;

celulacont (tipo opaco)

Contém campos:

- *contribuicao;*

- ponteiro para a próxima *celulacont.*

listacont (tipo opaco)

Contém campos:

- Sentinela para a primeiro e a última célula da lista;

A função *inicializarListaCont* não possui parâmetros. Ela aloca uma lista do tipo contribuição e faz as sentinelas que apontam para a primeira e a última posição da lista apontarem para NULL. Ou seja, ela cria uma lista vazia.

A função *inicializarCont* recebe como parâmetros o nome do editor e o nome do arquivo. Ela aloca um tipo contribuição e seus respectivos campos. Além disso ela define o status dessa contribuição como 1. O status será necessário no momento de imprimir a lista de contribuições e será discutido mais a fundo posteriormente.

A função *insereContribuicao* recebe como parâmetros uma lista de contribuições, o nome do editor, o nome do arquivo de contribuição e uma lista de editores. Após inicializar uma célula do tipo contribuição, esta é inserida na última posição da lista.

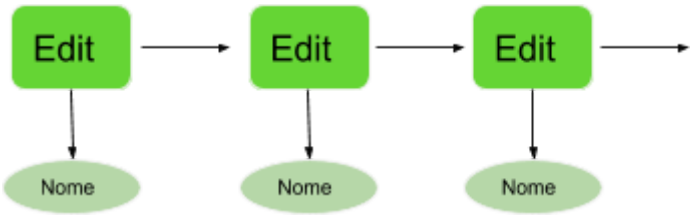
A função *retiraContribuicao* recebe como parâmetros uma lista de contribuições, o nome do editor, nome do arquivo de contribuição e nome da página. Com o nome do arquivo, é feita uma procura na lista de contribuições para encontrar a célula de contribuição correspondente. Caso ela seja encontrada, será verificado se o nome do editor dado corresponde ao nome do editor que a colocou na lista, pois somente este poderá "retirá-la" (o que não ocorre de fato, apenas seu campo de status que é mudado para zero). Caso a célula não exista ou um editor diferente do que a colocou na lista tente retirá-la, uma mensagem será escrita no arquivo de log.

A função *retiraContPeloEditor* recebe como parâmetros uma lista de contribuições e o nome de um editor. Essa função foi criada para tirar todas as contribuições de um determinado editor de uma lista de contribuições caso este seja retirado da lista de editores. A lista de contribuições será varrida e todas as contribuições que possuírem o campo de "nome_editor" igual ao editor dado como parâmetro serão retiradas.

A função *liberaCelulaCont* libera todo o espaço alocado pela célula dada como parâmetro. De forma análoga, a *liberaListCont* libera todo o espaço alocado por uma lista de contribuições(utilizando-se da liberaCelulaCont).

As funções *retornaLstContPrim*, *retornaContProx*, *retornaContNomeArq*, *retornaContNomeEditor* e *retornaContStatus* foram criadas para que pudéssemos acessar um determinado campo das structs definidas no arquivo de contribuição (contribuicao, celulacont e listacont) em outros arquivos.

EDITORES



No TAD editores criamos as structs que compõe a lista encadeada de editores: celulaEditor e listaeditor.

celulaEditor (tipo opaco)
Contém campos:
- nome do editor;
- ponteiro para a próxima célula;

listaeditor (tipo opaco)
Contém campos:
- sentinela para a primeiro e a última célula da lista;

A função *inicializarListaEditor* não possui parâmetros. Ela aloca a lista e faz com que as sentinelas que apontam para a primeira e a última posição da lista apontem para NULL. Em outras palavras, ela cria uma lista vazia.

A função *inicializarEditor* recebe como parâmetro apenas o nome do editor . Ela aloca uma célula do tipo editor e o espaço necessário para o campo do nome deste editor. Como o editor só precisa de uma string como “item”, decidimos por não criar um tipo editor de fato e colocamos a string para o nome do editor diretamente na célula.

A função *insereEditor* recebe como parâmetros uma lista de editores e o nome do editor. Após inicializar uma nova célula editor através da *inicializarEditor*, a função a insere na última posição da lista de editores.

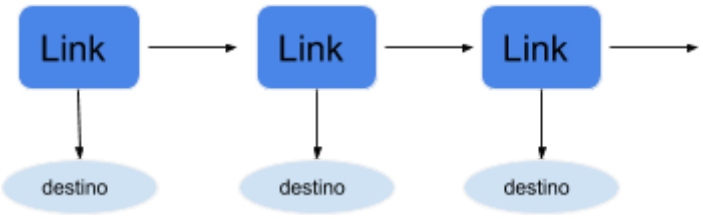
A função *liberaEditor* libera todo o espaço que foi alocado por uma célula editor dada como parâmetro. A função *liberaListEdit* funciona de maneira semelhante, liberando todas as células da lista de editores através da *liberaEditor*.

A função *existeEditor* recebe como parâmetros uma lista de editores e o nome de um editor. A função irá procurar na lista de editores por uma célula que

tenha o campo nome igual ao nome do editor, caso haja a função retornará 1 (um), caso contrário o retorno será igual a 0 (zero).

A função *retiraEditor* recebe como parâmetros uma lista de editores e o nome de um editor. Se for encontrado um editor com o mesmo nome que foi dado como parâmetro, ele será retirado da lista de editores.

LINKS



No TAD links criamos as structs que compõe a lista encadeada de links: celulalink e listalink.

celulalink (tipo opaco)
Contém campos:
- página destino;
- ponteiro para a próxima célula;

listalink (tipo opaco)
Contém campos:
- sentinela para a primeiro e a última célula da lista;

A função *inicializarListaLink* não possui parâmetros. Ela aloca a lista e faz com que as sentinelas que apontam para a primeira e a última posição da lista apontem para NULL.

A função *inicializarLink* recebe como parâmetro apenas o nome da página de destino. Ela aloca uma célula do tipo link e o espaço necessário para o campo do nome da página. Assim como a célula de um editor, o “item” da célula link está na própria célula, visto que se trata de apenas uma string.

A função *insereLink* recebe como parâmetros uma lista de links e o nome da página de destino. Após inicializar uma nova célula através da *inicializarLink*, a função a insere na última posição da lista.

A função *liberaCelulaLink* libera todo o espaço que foi alocado por uma célula link dada como parâmetro. A função *liberaListLinks* funciona de maneira parecida, liberando todas as células da lista de links através da função *liberaCelulaLink*.

A função *retiraLink* recebe como parâmetros uma lista de links, o nome da página de origem, o nome da página de destino e um número inteiro cod. Se for encontrado um link com o mesmo nome da página de destino, ele é retirado.

A variável cod passada como parâmetro foi criada com o intuito de controlar a mensagem n° 7 da função arqlog (ver TAD de log) que está presente na *retiraLink*. Quando uma página é retirada pela *retiraPagina*, os links para esta página também são retirados pela *retiraLinkAlternativa*. Caso não haja links para serem retirados a mensagem seria impressa no arquivo de log, o que seria indesejado. Assim na *retiraLinkAlternativa* o cod é passado como 0 (zero) para que a mensagem não seja escrita.

As funções *retornaLstLinkPrim* e *retornaLinkProx* foram criadas para que pudéssemos acessar um determinado campo das structs definidas no arquivo de links em outros arquivos. Elas simplesmente retornam o primeiro elemento de uma lista de links e a próxima célula, respectivamente.

PRINT

O TAD de print foi criado para armazenar todas as funções de impressão do trabalho, que são: *printHistCont*, *printCont*, *printLinks*, *imprimePagina* e *imprimewiked*. O arquivo não possui nenhuma struct, apenas funções.

A função *printHistCont* recebe como parâmetros uma célula página e o arquivo de uma página. Ela percorrerá a lista de contribuições desta página e imprimirá seu histórico no arquivo dado como parâmetro. Caso uma das contribuições possua o campo de status com o valor 0 (zero), a função imprimirá uma mensagem ao lado do nome da contribuição dizendo que ela foi retirada.

A função *printCont* recebe como parâmetros uma célula página e o arquivo de uma página. Toda a lista de contribuições dessa página será percorrida, aquelas que possuírem o status igual a 1 (um) serão selecionadas e arquivos serão abertos a partir do nome de arquivo que essas contribuições guardam. O conteúdo presente nesses arquivos será copiado caracter por caracter para o arquivo da página dado como parâmetro.

A função *printLinks* recebe como parâmetros uma célula página e o arquivo de uma página. A função percorrerá a lista de links da página em questão e imprimirá um campo informativo contendo todos os links que essa página possui.

A função imprimePagina recebe como parâmetros uma lista de páginas e o nome de uma página. Será realizada uma procura na lista de páginas e aquela que possuir o nome igual ao nome da página dado como parâmetro será selecionada. As funções *printHistCont*, *printCont* e *printLinks* serão chamadas e imprimirão as informações correspondentes a esta página.

A função *imprimewiked* recebe como parâmetro a lista de páginas. Ela utilizará a imprimePagina em cada célula página desta lista imprimindo, portanto, a lista de páginas inteira.

ARQUIVO DE LOG

O TAD de log possui apenas uma função: a *arqlog*. Assim como o TAD de print, ele não possui nenhuma struct.

A função *arqlog* recebe como parâmetros uma variável cod e duas strings. Essa função é responsável por imprimir mensagens de erro no arquivo de log e possui duas mensagens relacionadas a função caminho, que imprimem no arquivo uma mensagem dizendo se há ou não caminho (direto ou indireto) entre duas páginas (para mais informações sobre isso checar a explicação da função caminho no TAD de páginas).

A função possui diversos informes de erro diferentes que vão desde mensagens devido a requisição de uma função que não existe até mensagens que indicam se um editor que não é responsável por uma contribuição tentou retirá-la.

WikED

A função *main* está no arquivo da WikED. Como entrada deste programa, recebemos um arquivo texto contendo os comandos a serem executados, e verificamos se ele existe. Caso não exista, finalizamos a execução do programa. Caso existir, inicializamos a lista de páginas e a lista de editores com as funções *inicializarListaPagina* e *inicializarListaEditor*.

Após isso, utilizamos a variável *funcao* para ler do arquivo de entrada a função que deseja-se executar. Cada uma das funções é identificada por um código, que é a posição dela na matriz *funcoes.* E para descobrir este código utilizamos um loop, até que encontremos o índice correspondente.

A seguir usamos um switch case para fazer a leitura da quantidade correta de parâmetros para ser lida do arquivo de entrada e realizar a chamada da função adequada.

Enquanto não chegarmos até o final do arquivo texto dado como entrada ou até que a função *fim* seja chamada, esse processo se repetirá.

CONCLUSÃO

Durante a execução do trabalho foi possível perceber que apesar do tipo página, do tipo contribuição, do tipo link e do tipo editor se tratarem de estruturas diferentes, as funções que as envolvem possuem muitas semelhanças de modo que uma estrutura genérica, por exemplo, poderia ter sido utilizada neste trabalho. Contudo, optamos por não utilizar essa abordagem por falta de familiaridade com este tipo de implementação.

De modo geral, grande parte das funções implementadas se assemelham a exemplos vistos em aula ou já implementados para a resolução dos exercícios, com exceção da função *caminho,* que exigiu uma abordagem de pensamento diferente e precisou de certo tempo para ser finalizada, em especial pelo seu conceito recursivo, que está ligado a necessidade de avaliar links indiretos entre as páginas.

BIBLIOGRAFIA

Sites:
https://inf.ufes.br/~pdcosta/ensino/2018-1-estruturas-de-dados/

Livros:
Celes, Cerqueira e Rangel. Introdução a Estruturas de Dados, Editora Elsevier, 2004.