UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
ENGENHARIA DE COMPUTAÇÃO

Trabalho de Estrutura de Dados I
WikED!

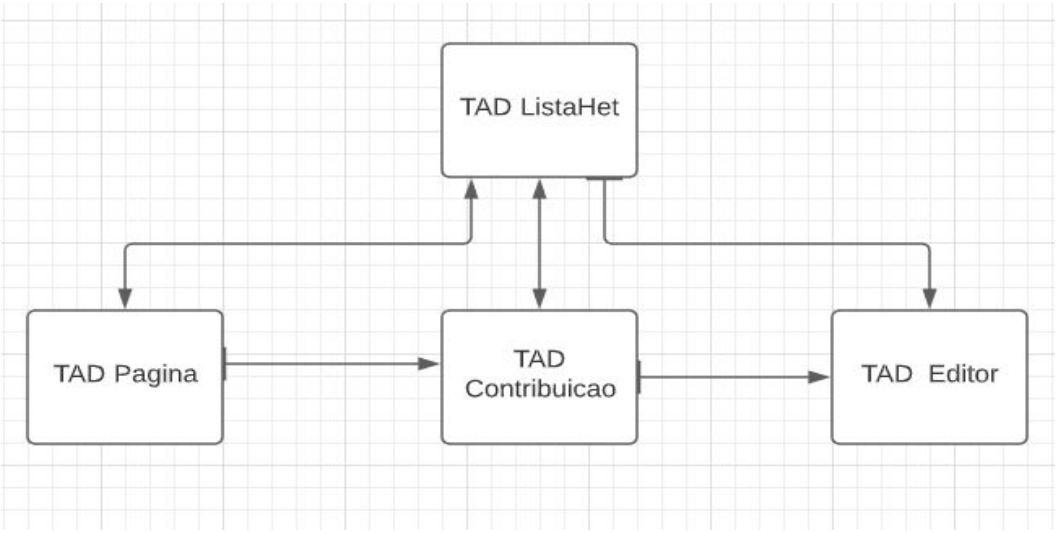
Mirelly Micaella da Silva

29 de novembro de 2020

Introdução

O trabalho proposto foi a implementação de um sistema que simula a construção colaborativa de uma enciclopédia similar à Wikipedia, chamada de WikED!.

A WikED! é definida por uma lista de páginas. Cada página possui uma lista de contribuições. Cada contribuição possui um editor.



Para o funcionamento do programa deve haver um diretório **./saida** no qual serão criados os arquivos das páginas e o arquivo de log. Foi criado um Makefile e para usá-lo basta digitar o comando **make gcc** que o programa será compilado e gerado o executável. Para executar o programa basta digitar **./wiked <diretório e nome do arquivo de entrada>**.

Ao fim da execução do programa serão gerados os seguintes arquivos dentro do diretório **./saida**:

- **<pagina>.txt** : arquivo contendo a impressão da página com o nome definido pelo arquivo de entrada. Será gerado um arquivo para cada página.

- **log.txt** : arquivo contendo os erros e resposta do comando caminho.

Metodologia

O programa principal funciona no arquivo main.c. Baseando-se no modelo de arquivo de entrada que foi disponibilizado, no funcionamento do programa foi criada uma string que armazena o comando e um vetor de strings que armazena as informações passadas após o comando. É usado o comando **do while** que tem como condição de parada o comando "FIM", assim os demais comando são lidos, verificados com o comando **if** e executados de acordo.

A seguir serão explicadas as principais funções do programa em seus respectivos TADs.

TAD ListaHet

Para este programa foi escolhido o uso de lista heterogênea com sentinela. A célula da lista contém um ponteiro do tipo *void* e um *int* tipo que pode ser umas das constantes definidas: **PAGINA**, **LINK** (também é o tipo página), **EDITOR** ou **CONTRIBUICAO**.

```
struct celula{
    Celula* prox;
    void* item;
    int tipo;
};

struct listaHet{
    Celula* Prim;
    Celula* Ult;
};
```

Este TAD é de muita importância para a implementação da wiked, portanto darei uma breve explicação do funcionamento das principais funções.

A função **InsereLista** retorna 0 se o item já está contido na lista, caso o item não esteja contido a função o isere e retorna 1.

A função **Retira** funciona igualmente para link, contribuição e editor, apenas a retirada de página é diferente. Para uma página ser retirada, primeiramente são verificadas todas as páginas da wiked e são retirados os links para a página a ser retirada. Em seguida a página é retirada da wiked e retornada para o cliente que pode destruí-la ou não.

A função **ImprimeWiked** imprime todas as páginas da wiked no diretório **./saida** .

As funções **ImprimeContribuicoesSimples** , **ImprimeContribuicoesCompletas** e **ImprimePaginasSimples** estão no plural pois recebem uma lista e imprime todos os itens chamando suas respectivas funções com o mesmo nome no singular.

A função **Busca** recebe uma lista, uma string com o nome que será usado como chave para a busca e um *int* tipo e retorna um *void** item correspondente a chave de busca.

A função **BuscaItem** se assemelha a função **Busca**, recebe uma lista, um *void** item e um *int* tipo e retorna 1 se o item está contido na lista e 0 se não está.

A função **LiberaLista** exclui apenas as células da lista, já a função **DestroiLista** destroi as células e os itens contidos na lista.

A função **Caminho** recebe uma página origem e uma página destino. Em seu funcionamento cria uma lista chamada mapa que serve para mapear as páginas que forem verificadas e envia esta lista para para a função **VerificaCaminhos**. Esta função, por sua vez, verifica todas os links da página origem adicionando as páginas verificadas no mapa, contudo por ser uma função recursiva ela chama a si mesma, porém passando a próxima página da lista de links como *paginaOrigem*. Quando todas as páginas tiverem sido verificadas, ou seja, quando a próxima página se verificada for NULL, a função **Caminho** verifica se a página destino apareceu no mapa, se sim ela retorna 1 e se não ela retorna 0.

TAD Editor

O TAD Editor é bem simples e possui apenas o nome do editor.

```
struct editor{
    char* nome;
};
```

TAD Contribuicao

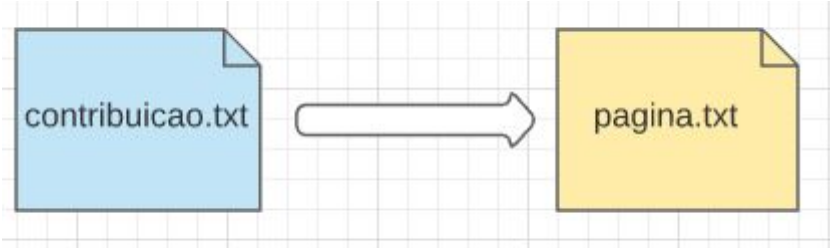
O TAD contribuição contém um tipo Editor e o nome do arquivo de texto do qual os textos da contribuição estão armazenados.

```
struct contribuicao{
    Editor* editor;
    char* arquivoTexto;
};
```

É importante explicar o funcionamento das funções que imprimem as contribuições.

A função **ImprimeContribuicaoSimples** imprime a contribuição no formato <nome do editor> <nome da contribuição> <retirada ou não>. Esta função recebe como parâmetro uma estrutura do tipo Contribuicao, o arquivo da página onde ela será printada e um *int* que pode ser 0 se aquela contribuição foi retirada e 1 ou qualquer outro número se ainda estiver ativa.

A função **ImprimeContribuicaoCompleta** recebe como parâmetro o arquivo da página que será imprimida e abre o arquivo com o texto da contribuição. A leitura do texto da contribuição é feita linha por linha usando a função **getline** da biblioteca *string.h.* Em seguida, a linha lida é printada no arquivo da página. Assim, todo o texto da contribuição é escrito na página.



TAD Pagina

```
struct pagina{
    char* nome;
    char* arquivo;
    ListaHet* contribuicoes;
    ListaHet* historicoContribuicoes;
    ListaHet* links;
};
```

A página possui uma lista de links que consiste em uma lista do tipo ListaHet onde são armazenados ponteiros do tipo Pagina, representando os links para outras páginas.

Além disso, possui duas listas de contribuições. A lista *historicoContribuicoes* contém todas as contribuições da página. A lista *contribuicoes* contém apenas as contribuições ativas na página, ou seja, que não foram retiradas.

Deste modo, para inserir uma contribuição na lista de contribuições da página não se usa a função **InsereLista** do TAD Lista diretamente. É chamada a função **AdicionaContribuicaoPagina** que verifica se contribuição já existe na página, se não existe, a função **InsereLista** é chamada para inserir a nova contribuição nas listas *historicoContribuicoes* e *contribuicoes* da página.

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
ENGENHARIA DE COMPUTAÇÃO

Conclusão

Este programa implementa todas as funções pedidas e seus possíveis erros são descritos no arquivo log.txt.

Foi muito difícil criar a função **Caminho,** pois não tenho muita experiência com recursão.
Além disso, por ter optado por usar lista heterogênea foi difícil tentar generalizar todas as funções no começo, mas no fim me ajudou bastante a simplificar o programa.

Fazer este trabalho me deu muito mais experiência no uso do Valgrind para encontrar erros.

29 de novembro de 2020

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
ENGENHARIA DE COMPUTAÇÃO

Referências bibliográficas

Grande parte do trabalho foi baseado nas aulas de ED I disponíveis no classroom. Porém, algumas referências para pequenas questões vieram de outros lugares.

Para lidar com arquivos usei como referência o meu próprio trabalho de PROG II.

Para encontrar erros usei este site que ensina a usar o Vlagrind: https://www.ic.unicamp.br/~rafael/materiais/valgrind.html

Para alguns erros mais pontuais fiz algumas pesquisas no site: https://stackoverflow.com/

29 de novembro de 2020