Universidade Federal do Espírito Santo - Departamento de Informática

1º Trabalho Prático - WikED!

Estruturas de Dados I (INF09292)
Período: 2020/1 EARTE
Profa Patrícia Dockhorn Costa
Aluno: Gabriel SIlva Simoura

Data: 28/11/2020

Introdução

Neste trabalho de Estruturas de Dados, o objetivo é implementar um sistema de construção colaborativa de uma enciclopédia, com funcionamento similar à *Wikipedia.*

De maneira que, nesses sistemas de construção colaborativa, os usuários contribuem para a construção de alguma estrutura que beneficiará toda uma comunidade. Ou seja, na plataforma WikED, os usuários finais são capazes de inserir e retirar contribuições das diversas páginas existentes no sistema.

TADs

1. **pagina.c / pagina.h :**

   Estrutura que poderão ser editadas pelos usuários, adicionando e retirando contribuições. No TAD pagina.c, são armazenados em sua estrutura, apenas ponteiros para:

   Nome da Página: Utilizado para identificação da página;

   Nome do Arquivo: Utilizado para manipular o conteúdo da página em si, adicionando e retirando, contribuições, lista de links e histórico de contribuições.

2. **editor.c / editor.h :**

   Estrutura que representa os usuários do programa, nomeados “editores”. No TAD editor.c, são armazenados em sua estrutura apenas um ponteiro para:
   Nome do Editor: Utilizado para identificação do usuário;

3. **contribuicao.c / contribuicao.h:**

   Estrutura que representa, como o próprio nome nos infere, a contribuição/colaboração de cada editor/usuário. Portanto, são pedaços de textos fornecidos pelos editores.
   No TAD contribuicao.c, são armazenados em sua estrutura:

   Nome do Arquivo: Um ponteiro para o nome do arquivo onde está armazenado o texto da contribuição, utilizado para inserção das contribuições nas páginas;

   Status: Variável tipo inteira na qual define a característica da contribuição, se ela foi retirada (*status = 1*) ou não retirada (*status = 0*);

4. **listaLink.c / listaLink.h:**

   Estrutura que representa uma lista de links presentes em cada página, ou seja, cada página existente na WikED, pode possuir uma lista de links que “levam” para outras páginas da plataforma. No TAD listaLink.c, há duas estruturas:

   CélulaLink: Corresponde a cada elemento da lista, nomeado de “célula”, nessa estrutura há um ponteiro para uma determinada página, assim como um ponteiro para a próxima celula da lista de links;

ListaLink: Corresponde à estrutura que aponta para cada página existente na lista de links de uma página específica. Possui dois ponteiros para a estrutura **CelulaLink,** apontando, respectivamente, para o início e para o fim da lista;

5. **listaContribuicao.c / listaContribuicao.h :**

Estrutura que representa uma lista de contribuições presentes em cada página, ou seja, cada página existente na WikED, pode possuir várias contribuições. No TAD listaContribuicao.c, há duas estruturas:

CélulaContribuicao: Corresponde a cada elemento da lista, nomeado de "célula", nessa estrutura há um ponteiro para um determinado editor, assim como um ponteiro para uma contribuição e para a próxima celula da lista de contribuições;

ListaContribuicao: Corresponde à estrutura que aponta para cada contribuição e seu editor/contibuinte de uma página específica. Possui dois ponteiros para a estrutura **CelulaContribuicao,** apontando, respectivamente, para o início e para o fim da lista;

6. **listaPagina.c / listaPagina.h:**

Estrutura que representa uma lista de Páginas, ou seja, uma lista com todas as páginas existentes no sistema WikED. No TAD listaPagina.c, há duas estruturas:

CelulaPagina: Corresponde a cada elemento da lista, nomeado de "célula", nessa estrutura há um ponteiros para as estruturas: Pagina, Lista de Links, Lista de Contribuiçoes e para a Próxima Célula da lista.

ListaPagina: Corresponde à estrutura que aponta para cada página e seus elementos existente, das páginas da plataforma Possui dois ponteiros para a estrutura **CelulaPagina,** apontando, respectivamente, para o início e para o fim da lista;

7. **listaEditor.c / listaEditor.h :**

Estrutura que representa uma lista de Editores, ou seja, uma lista com todos os editores existentes no sistema WikED. No TAD listaEditor.c, há duas estruturas:

CelulaEditor: Corresponde a cada elemento da lista, nomeado de "célula", nessa estrutura há um ponteiros para as estruturas: Editor, Lista de Contribuiçoes e para a Próxima Célula da lista.

ListaEditor: Corresponde à estrutura que aponta para cada editor e sua lista de contribuições das páginas da plataforma Possui dois ponteiros para a estrutura **CelulaEditor,** apontando, respectivamente, para o início e para o fim da lista;

Metodologia

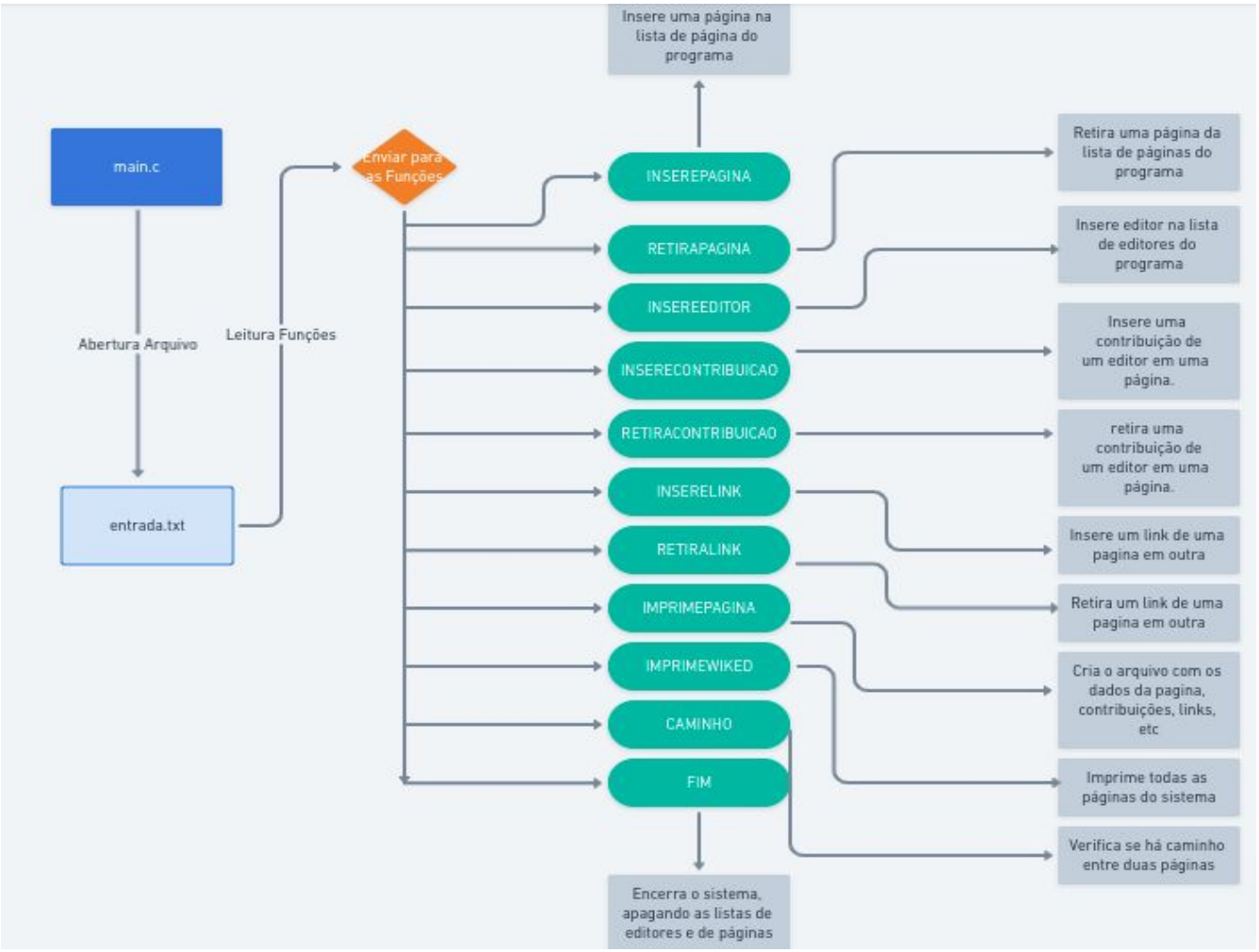
O Programa inicia-se na main, fazendo a abertura e em seguida a leitura do arquivo de “entrada,txt”.
Após a verificar se o arquivo foi aberto com sucesso é feito o seguinte tratamento de leitura de lixo no início do arquivo:

- Leio o arquivo até encontrar um caractere válido, quando encontro, insiro esse caracter válido na primeira posição do vetor da função que será executada posteriormente;
- Em seguida leio caracter por caracter para preencher o vetor para a primeira função a ser executada;

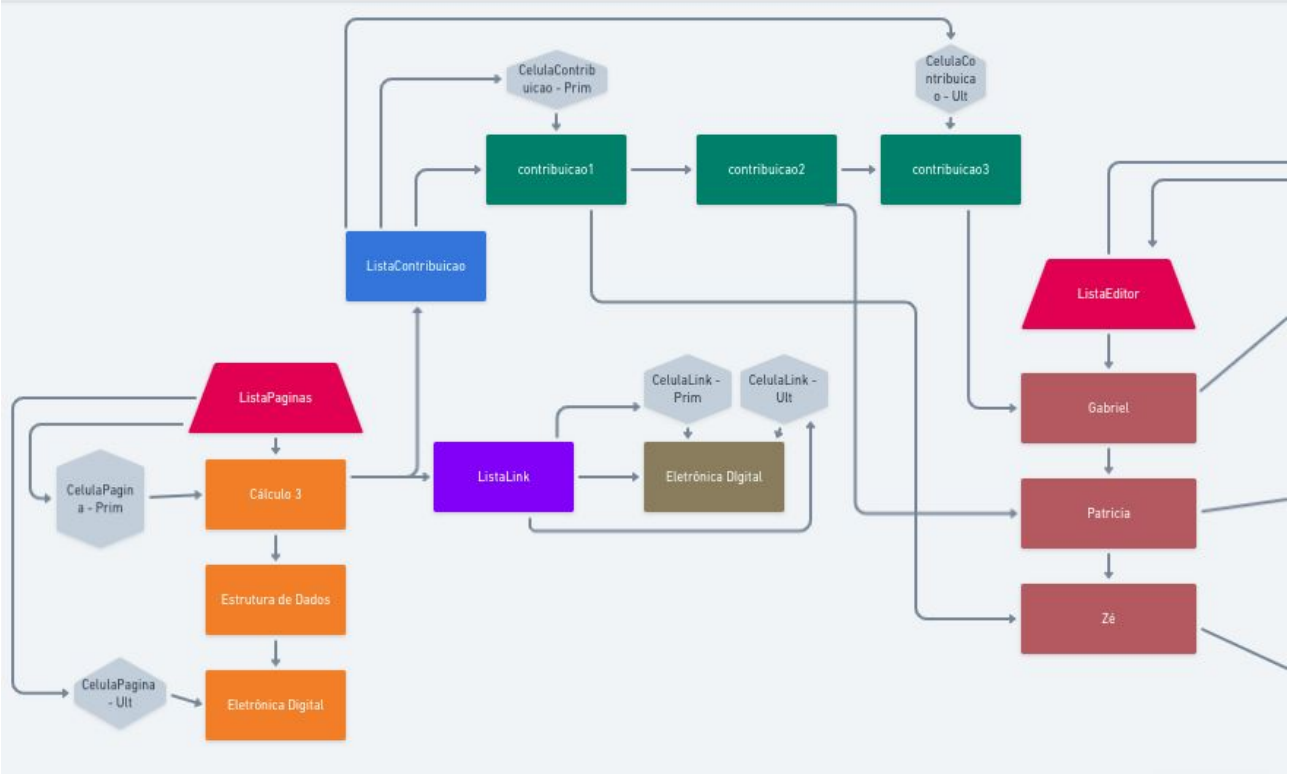
```
//Lê o arquivo até encontrar um char válido;
while (!(lixo >= 'A' && lixo <= 'Z') && !(lixo >= 'a' && lixo <= 'z'))
{
    fscanf(fileEntrada, "%c", &lixo);
}

//Atribuindo o primeiro char válido à primeira posição da string da função;
funcao[0] = lixo;

//Lê a primeira função do arquivo de entrada.txt;
for (int i = 1; lixo != ' '; i++)
{
    fscanf(fileEntrada, "%c", &lixo);

    if (lixo != ' '){ funcao[i] = lixo; }

    else { funcao[i] = '\0'; }
}
```

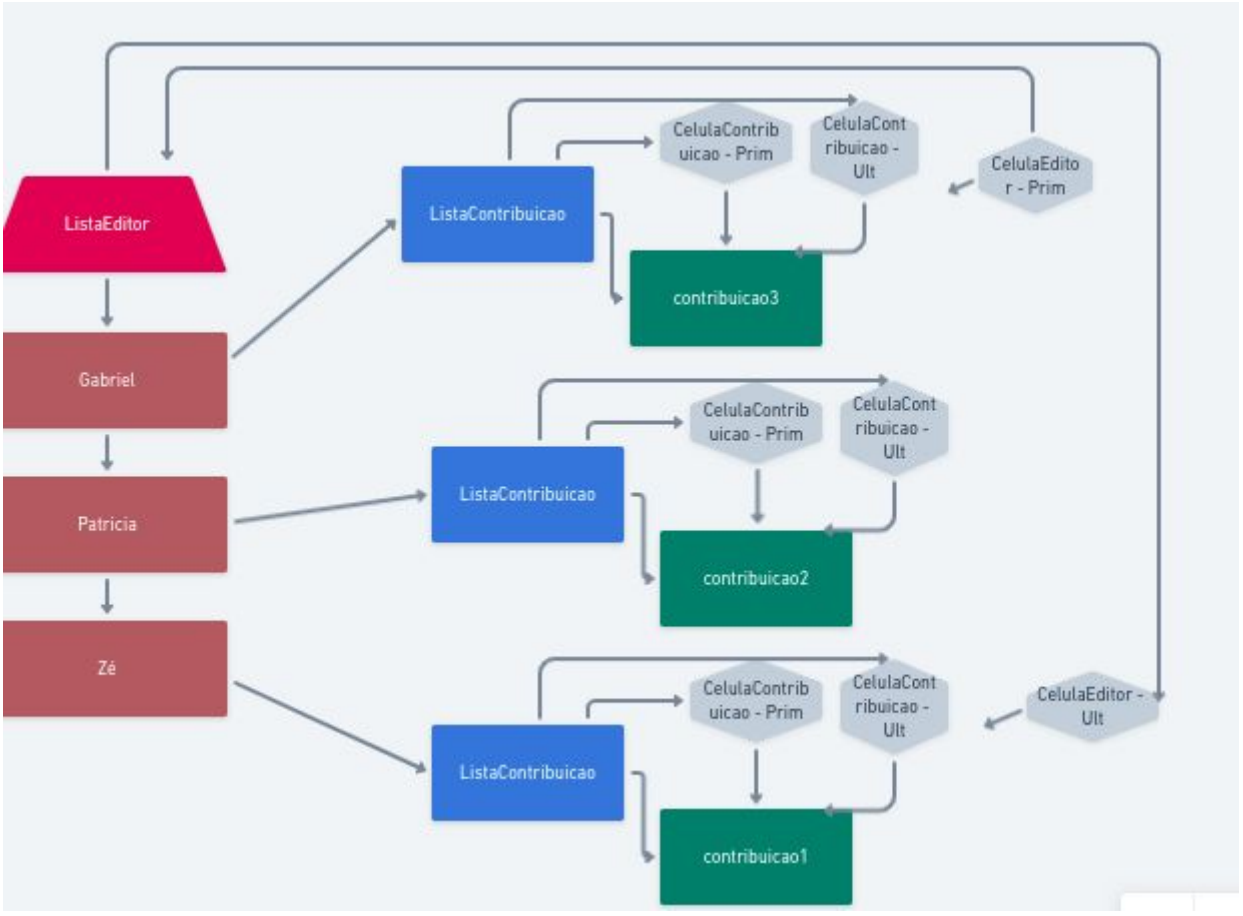
- Após isso, a cada função lida, é lido juntamente os parâmetros passados e “enviados” às suas respectivas funções, por exemplo: “INSEREPAGINA Fisica fisica.txt”;
  função: “INSEREPAGINA”;
  parametro1: “Fisica”;
  parametro2: “física.txt”;

1. Funcionamento main.c



2. Estruturação do programa WikED (Exemplo)i:





3. Principais Funções:

CAMINHO: Após a verificar se as duas Páginas, passadas por parâmetro, existem e não são iguais, inicializo uma lista de links de páginas percorridas entre as duas páginas e uso a função que preenche essa lista.

```
ListaLink *paginasPercorridas = IniciaListaLink();

CaminhoListaLink(listaPagina, paginasPercorridas, paginaAtual);
```

A função “CaminhoListaLink” realiza o seguinte procedimento:
Percorre cada página da lista de link de cada página da lista de links da página atual e vai adicionando-as, à lista de páginas percorridas, ao final, se a página de destino está contida na lista de páginas percorridas, logo, há caminho entre as duas páginas.

```
//Insere a pagina atual na lista de links das paginas percorridas;
InsereListaLink(paginasPercorridas, paginaAtual);

//Retorna lista de links da página atual;
ListaLink *novaListaLink = RetornaListaLinkListaPagina(listaPagina, RetornaNomePagina(paginaAtual));
CelulaLink *celulaAtual;

//Faz a busca
for (celulaAtual = novaListaLink->Prim; celulaAtual != NULL; celulaAtual = celulaAtual->proxima)
{
    //Se não encontrar a pagina na lista de percorridas, então ela é adicionada;
    if (RetornaCelulaLinkListaLink(paginasPercorridas, RetornaNomePagina(celulaAtual->pagina)) == NULL)
    {
        CaminhoListaLink(listaPagina, paginasPercorridas, celulaAtual->pagina);
    }
}
```

4. **Makefile**
   a. ***make wiked:*** compila todos os arquivos (.c) do programa, possui alguns parâmetros que alertam, por exemplo, se há variáveis não utilizadas;

```
wiked: main.c contribuicao.c editor.c listaLink.c pagina.c listaEditor.c listaPagina.c
    gcc -ggdb -std=c99 -Wall -Werror -o wiked.out *.c
```

   b. ***make clean:*** apaga o último wiked.out compilado:

```
clean:
    rm -f *.o a.out core wiked.out
```

   c. ***make test:*** testes feitos com o valgrind, verificar vazamento de memória:

```
test:
    valgrind --leak-check=full --show-leak-kinds=all ./wiked.out entrada.txt -s
```

5. **Principais Tratamentos de Erros Omitidos, Realizados:**

   a. Na função Insere Contribuição:
      i. Verifica se a Página existe;
      ii. Verifica se o Editor existe;
      iii. Verifica se a contribuição já foi adicionada previamente;

   b. Na função Retira Contribuição:
      i. Verifica se a Página existe;
      ii. Verifica se o Editor existe;
      iii. Verifica se a contribuição que deseja retirar existe na Página;
      iv. Verifica se a contribuição já foi retirada anteriormente da Página;
      v. Verifica se o Editor pode retirar tal contribuição;

   c. Na função main.c
      i. Caso o usuário não passe o arquivo entrada.txt;
      ii. Caso o usuário passe o arquivo com nome diferente de entrada.txt

   d. Na função Caminho:
      i. Se o usuário buscar caminho entre páginas inexistentes;
      ii. Se o usuário buscar caminho entre a mesma página;

Conclusões

O trabalho de Estrutura de dados foi de suma importância para aplicar os conceitos de listas encadeadas com sentinelas, conceitos estes ensinados pela professora nas aulas síncronas tanto quanto, assíncronas através das aulas gravadas.
MInha maior dificuldade, foi o princípio do trabalho, na forma de “o que cada TAD é composto? “, “devo ler os arquivos de contribuição e armazenar suas informações?”, “as páginas podem ter duas contribuições iguais?”. No entanto, à medida que o trabalho foi sendo desenvolvido, as dúvidas iniciais foram sanadas e algumas mais complexas apareceram, “como implementar a função caminho sem cair em um loop infinito?.
Em suma, pode se inferir que, com a ajuda das aulas síncronas e de alguns sites pesquisados, pude aplicar as principais funções do sistema WikED, não havendo nenhum vazamento de memória na aplicação dos casos testes.

Referências Bibliográficas:

- https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/reach.html