UNISINOS - UNIVERSIDADE DO VALE DO RIO DOS SINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS – Curso: Informática

LABORATÓRIO II

Disciplina: Linguagem de Programação PASCAL **E-mail:** ***osorio@exatas.unisinos.br***
Professor responsável: ***Fernando Santos Osório*** **Web:**
Semestre: 2004/1 ***http://www.inf.unisinos.br/~osorio/lab2.html***
Horário: 41 **Xerox : *Pasta 54* – LAB. II (Xerox do C6/6)**

TRABALHO PRÁTICO 2004/1 – GRAU B (Versão 1.0 – 18/06/2004)

Faça um programa para ler um arquivo em HTML (ou mesmo XML) e gerar uma ***árvore genérica*** representando este arquivo, onde a seguir serão realizadas algumas consultas, de acordo com o especificado mais abaixo. O programa deve possuir um menu com as seguintes opções, descritas logo a seguir: ler arquivo, verificar integridade das TAGs, localizar e salvar TAGs, localizar e salvar texto comum, localizar string e exibir o contexto, salvar arquivo sem as TAGs, salvar árvore genérica com identação, exibir estatísticas, e opcionalmente fazer um "browser" para TAGs HTML (ponto extra).

1. **Ler o arquivo** de dados (*.htm, *.xml ou *.txt) do disco: Esta opção permite ler do disco um arquivo em formato texto, contendo "tags" (exemplo: <TAG> e </TAG>), bem como textos "comuns". Este arquivo deve ser lido usando uma função "proximo" escrita em Pascal que retorna a próxima string do arquivo cada vez que é chamada, onde esta função é fornecida pelo professor (conhecida como função "geradora de tokens"). Os strings lidos podem ser de 2 tipos: string comum de texto ou string do tipo TAG, onde este último tipo de string ainda pode ser do tipo início de TAG ou final de TAG. Os dados lidos devem ser armazenados em uma árvore genérica conforme exemplo abaixo.

   Exemplo de arquivo HTML de entrada:

```
<HTML>
<HEAD>
<TITLE>Titulo da Pagina</TITLE>
</HEAD>
<BODY>
<H1>Cabeçalho 1</H1>
Texto da Página
</BODY>
</HTML>
```

   Exemplo de modo de uso e funcionamento da rotina "próximo":

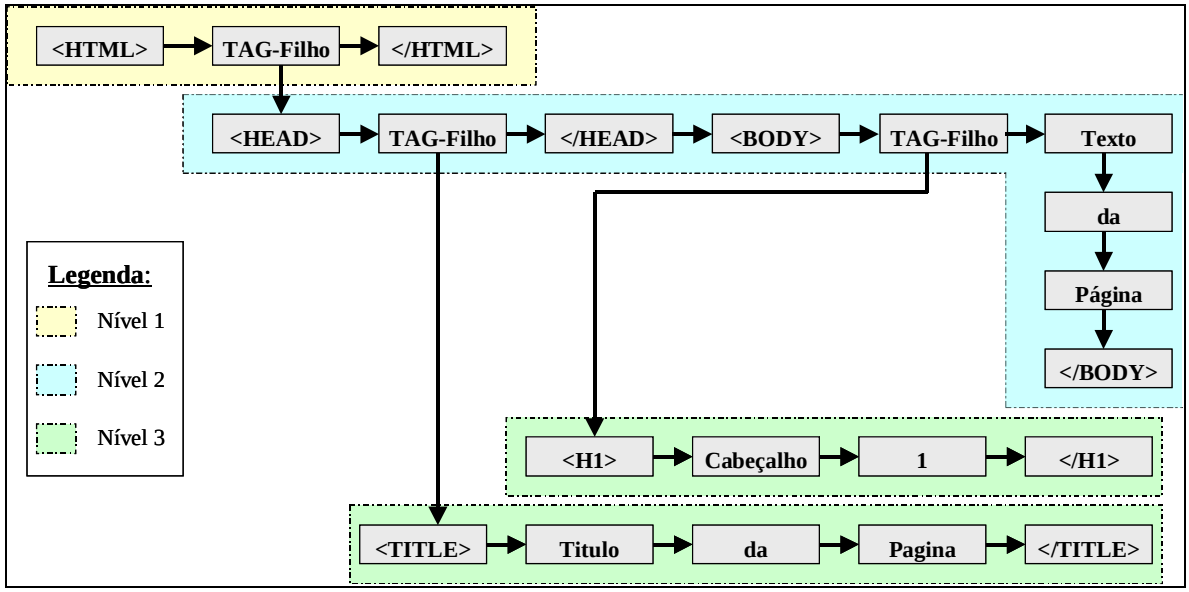
```
While  Proximo(arqtxt, texto) = True
Do  { ... seu código ... }
```

   A variável "texto" irá retornar os seguintes dados, onde caso seja encontrado o final do arquivo é retornado FALSE pela função, e caso contrário é sempre retornado TRUE:

Function Proximo (var arqtxt:text; var texto: string): boolean;

Texto = { “<HTML>”, “<HEAD>”, “<TITLE>”, “Titulo”, “da”, “Pagina”, “</TITLE>”, “</HEAD>”, “<BODY>”, “<H1>”, “Cabeçalho”, “1”, “</H1>”, “Texto”, “da”, “Página”, “</BODY>”, “</HTML>” }

Exemplo da árvore genérica a ser gerada: (árvore que espera-se obter ao ler o arquivo exemplo)



Se você quiser saber mais sobre HTML, consulte:

http://werbach.com/barebones/barebone_port.html
http://www.htmlhelp.com/reference/html40/
http://www.htmlcodetutorial.com/

2. Verificar a integridade das TAGs: esta opção deve verificar se toda a TAG que é aberta é também posteriormente fechada (dentro de seu respectivo nível). No exemplo acima podemos constatar que todas as TAGs iniciam (<TAG>) e depois são terminadas de modo correto (</TAG>). Caso seja encontrado algum erro, o programa deve indicar qual é a TAG com problema, exibindo o texto desta TAG. Os erros (mensagem de “***warning”***) possíveis são:
   - Uma TAG que inicia mas que não termina;
   - Uma TAG que termina sem no entanto ter iniciado;

   OBS: Esta opção é opcional caso os erros sejam detectados (e exibidos) na leitura do arquivo.

3. Localizar e salvar TAGs em disco: esta opção deve procurar todas as ocorrências de uma string que esteja contida em alguma TAG (busca uma substring na TAG), salvando em disco as TAGs que contém esta string. O usuário informa o texto da string a ser procurada dentro de alguma TAG, bem como o nome do arquivo onde deverão ser salvas as TAG encontradas contendo a string procurada. Exemplo de interação com o usuário:

```
>> OPCAO: Procura TAG contendo o texto especificado <<
> Nome do arquivo: mails.txt
> Texto procurado nas TAGs: mailto
> Procurando texto... Encontradas 10 ocorrências da string
> Arquivo com TAGs foi gravado!
```

4. Localizar e salvar "texto comum" em disco: esta opção deve procurar todas as ocorrências de uma string que ***não*** esteja contida em alguma TAG, salvando em disco as palavras que contém esta string (busca substring). O usuário informa o texto da string a ser procurada, bem como o nome do arquivo onde deverão ser salvas as palavras encontradas contendo a string procurada. Exemplo de interação com o usuário:

```
>> OPCAO: Procura texto comum contendo o texto especificado <<
> Nome do arquivo: filtro.txt
> Texto procurado: sex
> Procurando texto... Encontradas 3 ocorrências da string
> Arquivo com  texto procurado foi gravado!
```

5. Localizar uma string qualquer procurada, dentro ou fora de alguma TAG: esta opção deve procurar a primeira ocorrência de uma string, contida em algum elemento do arquivo HTML (TAG ou texto comum). Exibir na tela o texto ou TAG completo que contém a string procurada, indicando o contexto onde foi encontrada. O contexto é uma lista de todos os TAGs que antecedem na hierarquia da árvore este texto. Exemplo de interação com o usuário:

```
>> OPCAO: Procura o texto especificado <<
> Texto procurado: @
> Procurando texto...
> Encontrado:  mailto:fulano@servidor.com.br
> Contexto: <HTML> <BODY> <A HREF="mailto:fulano@servidor.com.br">
```

6. Salvar a árvore em disco sem as TAGs: esta opção deve gravar em disco um arquivo texto contendo apenas o "texto comum" do documento HTML, sem as TAGs. O usuário informa o nome do arquivo e o programa gera um arquivo em disco contendo todos os elementos da árvore, na ordem original em que aparecem no texto, sem as TAGs.
7. Salvar a árvore genérica com identação: esta opção deve gravar em disco um arquivo texto com todo o conteúdo da árvore. Os elementos da árvore devem ser apresentados um em cada linha do arquivo texto, onde devem estar identados com um deslocamento de 2 espaços em branco para cada novo nível da árvore. Exemplo de um arquivo gerado:

```
<HTML>
  <HEAD>
    <TITLE>
    Título
    da
    Pagina
    </TITLE>
  </HEAD>
  <BODY>
    <H1>
    Cabeçalho
    1
    </H1>
  <BODY>
  Texto
  da
  Página
</HTML>
```

8. Exibir estatísticas sobre o documento HTML: esta opção deve exibir na tela algumas estatísticas sobre o documento, a saber, o número total de TAGs (início de TAG), o número de nodos de texto comum, a profundidade máxima da árvore e o número total de nodos. Para o exemplo adotado neste texto, teremos:

    - Número de TAGs: **5**
    - Número de nodos de texto comum: **8**
    - Profundidade da árvore: **3**
    - Número total de Nodos: **21**

9. Entrar no modo de navegação (browse) na árvore. Neste modo, o usuário irá visualizar sempre apenas um nível da árvore, podendo navegar usando o teclado: tecla "A" = nodo anterior, tecla "S" = nodo seguinte, tecla "E" = Entra uma nível mais abaixo (se o nodo for do tipo tag-filho) e tecla "R" = Retorna para o nível de cima do atual. Os dados de um mesmo nível deverão ser apresentados na tela com um nodo em cada linha, com algum tipo de marca que permita identificar o "nodo corrente" e navegar entre os nodos. Sempre que for mudado de nível, deverão ser apagados os dados da tela e então listados somente os nodos do nível atual. Caso seja teclada a tecla "X", devemos sair do modo de navegação e voltar ao menu (***ponto extra – item opcional do trabalho***).

10. Sair do programa.

ATENÇÃO: As estruturas de dados devem ser baseadas (similares) as rotinas de manipulação de estruturas de dados que estudamos e implementamos na nossa disciplina. **Fazer um programa MODULAR e SEM USAR VARIÁVEIS GLOBAIS.**

ATENÇÃO:

- Os arquivos HTML podem ter TAGs que possuem início e não tem uma marca de fim, como por exemplo "<IMG SRC="imagem.jpg"> ou <HR>, por isso, nestes casos apenas um "warning" deve ser gerado, mas o programa deve continuar a funcionar normalmente.

- Algumas TAGs também aceitam parâmetros, como por exemplo <A HREF="home.html"> ou <P ALIGN="center"> e a sua TAG final correspondente é composta apenas do elemento principal, como o </A> e o </P>. Procure levar isto em consideração, o que será discutido em aula com o professor. Além disto, os arquivos HTML (ou XML) usados para testar o programa implementado serão previamente disponibilizados, sendo arquivos HTML mais simples.

INFORMAÇÕES COMPLEMENTARES:
Consulte a página da disciplina na Internet para obter o programa Pascal (Unit) que implementa a função "proximo" => http://inf.unisinos.br/~osorio/lab2.html