AED 1 - Trabalho 3 - Validação HTML

Bruno Tomé[1], **Cláudio Menezes**[1]

[1]Instituto Federal de Minas Gerais (IFMG)
São Luiz Gonzaga, s/nº - Formiga / MG - Brasil

ibrunotome@gmail.com, claudiomenezio@gmail.com

Abstract. *Validating a HTML source with cell structure.*

Resumo. *Validando HTML com a estrutura pilha.*

1. Introdução

A proposta de realização deste trabalho é tratar 8 tags de arquivos HTML, verificar se as tags estão casadas (abrindo e fechando em quantidade igual). A estrutura usada é a pilha, visto que a medida em que o caractere é encontrado, um valor numérico é adicionado a pilha, quando o caractere de fechamento da tag é encontrado, desempilhamos. Ao fim do arquivo HTML para cada tag, se a pilha estiver vazia significa que o documento está correto, se a pilha não estiver vazia significa que uma tag não foi fechada, e o programa informa em qual linha contém o erro para facilitar a corretude do HTML.

2. Implementação

2.1. Descrição das estruturas de dados usadas no programa

A estrutura utilizada foi pilha, como explicado na introdução, quando encontramos uma tag html especificada empilhamos um inteiro (não há a necessidade de empilhar o caractere, escolhi o inteiro apenas para controle), se encontramos o fechamento dessa tag desempilhamos. Ao final, se a pilha não estiver vazia, significa que abriu uma tag mas não fechou, então o programa retornará que há um erro na linha em que essa tag abriu.

2.2. Funcionamento das principais funções e procedimentos utilizados

Primeiramente, verifico se o número de argumentos passados no terminal está correto, deve ser passado junto ao programa um arquivo html.

Após essa parte faço a leitura do arquivo html, linha a linha, incremento um contador de linhas e chamo os procedimentos que verificam através de uma pilha cada uma das tags (criei uma para cada uma das 8 tags que devem ser tratadas). Após esta parte chamo os procedimentos que verificam se a pilha está vazia, através de uma flag, se estiver fazia, não existe erro algum, então a flag continua baixa e o texto dizendo que o arquivo html está correto é impresso na tela. Se a pilha não estiver vazia a flag sobe, e um texto mostrando em qual linha está o erro é impresso no console.

2.3. Formato de entrada e saída dos dados

2.3.1. Formato de entrada

Arquivo html passado como parâmetro no console.

2.3.2. Formato de saída

Resultado impresso no console.

2.4. Como executar o programa

No terminal com o arquivo do código fonte no Desktop digite o seguinte código:

cd Desktop

fpc corpo.pas -ocorpo.bin

./corpo.bin <ARQUIVO.HTML>

3. Conclusão

O trabalho contribuiu para afirmar ainda mais o que eu já tinha certeza, que a pilha é uma ótima estrutura e é a mais fácil de se utilizar/implementar. Não houve dúvidas em relação a implementação, apenas tive uma dúvida que foi sanada por email, se era para tratar realmente um arquivo html que ”funcione”ou se era apenas para tratar as 8 linhas de tags do exemplo.

4. Bibliografia

Apenas o próprio enunciado e a biblioteca pilha passada pelo professor.

5. Listagem do código fonte

5.1. Corpo

```
1  {
2      Bruno Tome - 0011254 - ibrunotome@gmail.com
3      Claudio Menezes - 0011255 - claudiomenezio.@gmail.com
4
5      O lowercase converte os caracteres maiusculos, se houver, para
           minusculo
6      cumprindo com o requisito de trabalhar com maiusculo e minusculo.
7  }
8
9  program tabalhopilhahtml;
10
11 uses biblioteca, crt;
12
13 procedure verificahtml(var pilhahtml : pilha; s : string; numlinha :
       byte; var errohtml : byte);
14 var i : byte;
15 begin
16     // Empilha html
17     for i:=1 to length(s) do
18     begin
19          if ((s[i] = '<') and (lowercase(s[i+1]) = 'h') and ((lowercase
               (s[i+2])) = 't') and ((lowercase(s[i+3])) = 'm') and ((
               lowercase(s[i+4])) = 'l') and ((lowercase(s[i+5])) = '>'))
                then
20          begin
```

```
21              inserir(pilhahtml,1); // empilha um inteiro so para
                    controle, nao ha necessidade de empilhar o caractere
22              errohtml := numlinha; // salva o numero da linha, caso
                    exista o erro um procedimento ira imprimi-lo depois
23          end;
24
25          // Desempilha html
26          if ((s[i] = '<') and (lowercase(s[i+1]) = '/') and (lowercase(
                s[i+2]) = 'h') and ((lowercase(s[i+3])) = 't') and ((
                lowercase(s[i+4])) = 'm') and ((lowercase(s[i+5])) = 'l')
                and ((lowercase(s[i+6])) = '>')) then
27          begin
28              remover(pilhahtml);
29          end;
30      end;
31  end;
32
33  procedure verificahead(var pilhahead : pilha; s : string; numlinha :
        byte; var errohead : byte);
34  var i : byte;
35  begin
36      // Empilha head
37      for i:=1 to length(s) do
38       begin
39          if ((s[i] = '<') and ((lowercase(s[i+1]) = 'h')) and ((
                lowercase(s[i+2])) = 'e') and ((lowercase(s[i+3])) = 'a')
                and ((lowercase(s[i+4])) = 'd') and ((lowercase(s[i+5])) =
                 '>')) then
40          begin
41              inserir(pilhahead,1); // empilha um inteiro so para
                    controle, nao ha necessidade de empilhar o caractere
42              errohead := numlinha; // salva o numero da linha, caso
                    exista o erro um procedimento ira imprimi-lo depois
43          end;
44
45          // Desempilha head
46          if ((s[i] = '<') and (s[i+1] = '/') and ((lowercase(s[i+2])) =
                 'h') and ((lowercase(s[i+3])) = 'e') and ((lowercase(s[i
                +4])) = 'a') and ((lowercase(s[i+5])) = 'd') and ((
                lowercase(s[i+6])) = '>')) then
47          begin
48              remover(pilhahead);
49          end;
50       end;
51  end;
52
53  procedure verificatitle(var pilhatitle : pilha; s : string; numlinha :
        byte; var errotitle : byte);
54  var i : byte;
55  begin
56      // Empilha title
57      for i:=1 to length(s) do
58       begin
59          if ((s[i] = '<') and (lowercase(s[i+1]) = 't') and ((lowercase
                (s[i+2])) ='i') and ((lowercase(s[i+3])) = 't') and ((
                lowercase(s[i+4])) = 'l') and ((lowercase(s[i+5])) = 'e')
```

```
            and ((lowercase(s[i+6])) = '>')) then
60          begin
61              inserir(pilhatitle,1); // empilha um inteiro so para
                    controle, nao ha necessidade de empilhar o caractere
62              errotitle := numlinha; // salva o numero da linha, caso
                    exista o erro um procedimento ira imprimi-lo depois
63          end;
64
65         // Desempilha title
66          if ((s[i] = '<') and (s[i+1] = '/') and ((lowercase(s[i+2])) =
                't') and ((lowercase(s[i+3])) = 'i') and ((lowercase(s[i
               +4])) = 't') and ((lowercase(s[i+5])) = 'l') and ((
               lowercase(s[i+6])) = 'e') and ((lowercase(s[i+7])) = '>'))
                then
67          begin
68              remover(pilhatitle);
69          end;
70      end;
71  end;
72
73  procedure verificabody(var pilhabody : pilha; s : string; numlinha :
        byte; var errobody : byte);
74  var i : byte;
75  begin
76      // Empilha body
77      for i:=1 to length(s) do
78      begin
79          if ((s[i] = '<') and (lowercase(s[i+1]) = 'b') and ((lowercase
                (s[i+2])) = 'o') and ((lowercase(s[i+3])) = 'd') and ((
               lowercase(s[i+4])) = 'y') and ((lowercase(s[i+5])) = '>'))
                then
80          begin
81              inserir(pilhabody,1); // empilha um inteiro so para
                    controle, nao ha necessidade de empilhar o caractere
82              errobody := numlinha; // salva o numero da linha, caso
                    exista o erro um procedimento ira imprimi-lo depois
83          end;
84
85         // Desempilha body
86          if ((s[i] = '<') and (s[i+1] = '/') and ((lowercase(s[i+2])) =
                'b') and ((lowercase(s[i+3])) = 'o') and ((lowercase(s[i
               +4])) = 'd') and ((lowercase(s[i+5])) = 'y') and ((
               lowercase(s[i+6])) ='>')) then
87          begin
88              remover(pilhabody);
89          end;
90      end;
91  end;
92
93  procedure verificah1(var pilhah1 : pilha; s : string; numlinha : byte;
        var erroh1 : byte);
94  var i : byte;
95  begin
96      // Empilha h1
97      for i:=1 to length(s) do
98      begin
```

```
99          if ((s[i] = '<') and (lowercase(s[i+1]) = 'b') and ((lowercase
                (s[i+2])) = 'o') and ((lowercase(s[i+3])) = 'd') and ((
                lowercase(s[i+4])) = 'y') and ((lowercase(s[i+5])) = '>'))
                 then
100         begin
101             inserir(pilhah1,1); // empilha um inteiro so para controle
                    , nao ha necessidade de empilhar o caractere
102             erroh1 := numlinha; // salva o numero da linha, caso
                    exista o erro um procedimento ira imprimi-lo depois
103         end;
104
105        // Desempilha h1
106         if ((s[i] = '<') and (s[i+1] = '/') and ((lowercase(s[i+2])) =
                'b') and ((lowercase(s[i+3])) = 'o') and ((lowercase(s[i
                +4])) = 'd') and ((lowercase(s[i+5])) = 'y') and ((
                lowercase(s[i+6])) ='>')) then
107         begin
108             remover(pilhah1);
109         end;
110     end;
111 end;
112
113 procedure verificacenter(var pilhacenter : pilha; s : string; numlinha
        : byte; var errocenter : byte);
114 var i : byte;
115 begin
116     // Empilha center
117     for i:=1 to length(s) do
118     begin
119          if ((s[i] = '<') and (lowercase(s[i+1]) = 'c') and ((lowercase
                (s[i+2])) ='e') and ((lowercase(s[i+3])) = 'n') and ((
                lowercase(s[i+4])) = 't') and ((lowercase(s[i+5])) = 'e')
                and ((lowercase(s[i+6])) = 'r') and ((lowercase(s[i+7])) =
                '>')) then
120         begin
121             inserir(pilhacenter,1); // empilha um inteiro so para
                    controle, nao ha necessidade de empilhar o caractere
122             errocenter := numlinha; // salva o numero da linha, caso
                    exista o erro um procedimento ira imprimi-lo depois
123         end;
124
125        // Desempilha center
126         if ((s[i] = '<') and (lowercase(s[i+1]) = '/') and (lowercase(
                s[i+2]) = 'c') and ((lowercase(s[i+3])) ='e') and ((
                lowercase(s[i+4])) = 'n') and ((lowercase(s[i+5])) = 't')
                and ((lowercase(s[i+6])) = 'e') and ((lowercase(s[i+7])) =
                 'r') and ((lowercase(s[i+8])) ='>')) then
127         begin
128             remover(pilhacenter);
129         end;
130     end;
131 end;
132
133 procedure verificaol(var pilhaol : pilha; s : string; numlinha : byte;
        var errool : byte);
134 var i : byte;
```

```
135 begin
136     // Empilha ol
137     for i:=1 to length(s) do
138     begin
139          if ((s[i] = '<') and (lowercase(s[i+1]) = 'o') and (lowercase(
                s[i+2]) = 'l') and (lowercase(s[i+3]) = '>')) then
140          begin
141              inserir(pilhaol,1); // empilha um inteiro so para controle
                     , nao ha necessidade de empilhar o caractere
142              errool := numlinha; // salva o numero da linha, caso
                     exista o erro um procedimento ira imprimi-lo depois
143          end;
144
145         // Desempilha ol
146          if ((s[i] = '<') and (s[i+1] = '/') and ((lowercase(s[i+2])) =
                 'o') and (lowercase(s[i+3]) = 'l') and (lowercase(s[i+4])
                 = '>')) then
147          begin
148              remover(pilhaol);
149          end;
150      end;
151 end;
152
153 procedure verificali(var pilhali : pilha; s : string; numlinha : byte;
        var erroli : byte);
154 var i : byte;
155 begin
156     // Empilha li
157     for i:=1 to length(s) do
158     begin
159          if ((s[i] = '<') and (lowercase(s[i+1]) = 'l') and (lowercase(
                s[i+2]) = 'i') and (lowercase(s[i+3]) = '>')) then
160          begin
161              inserir(pilhali,1); // empilha um inteiro so para controle
                     , nao ha necessidade de empilhar o caractere
162              erroli := numlinha; // salva o numero da linha, caso
                     exista o erro um procedimento ira imprimi-lo depois
163          end;
164
165         // Desempilha li
166          if ((s[i] = '<') and (s[i+1] = '/') and ((lowercase(s[i+2])) =
                 'l') and (lowercase(s[i+3]) = 'i') and (lowercase(s[i+4])
                 = '>')) then
167          begin
168              remover(pilhali);
169          end;
170      end;
171 end;
172
173 // Inicio das verificacoes de erro
174
175 procedure verificaerrohtml(pilhahtml : pilha; var flag : boolean;
        errohtml : byte);
176 begin
177     if (not(vazia(pilhahtml))) then
178     begin
```

```
179         flag := true;
180         writeln('O Html contem um erro na linha ', errohtml);
181     end;
182 end;
183
184 procedure verificaerrohead(pilhahead : pilha; var flag : boolean;
        errohead : byte);
185 begin
186     if (not(vazia(pilhahead))) then
187     begin
188         flag := true;
189         writeln('O Html contem um erro na linha ', errohead);
190     end;
191 end;
192
193 procedure verificaerrotitle(pilhatitle : pilha; var flag : boolean;
        errotitle : byte);
194 begin
195     if (not(vazia(pilhatitle))) then
196     begin
197         flag := true;
198         writeln('O Html contem um erro na linha ', errotitle);
199     end;
200 end;
201
202 procedure verificaerrobody(pilhabody : pilha; var flag : boolean;
        errobody : byte);
203 begin
204     if (not(vazia(pilhabody))) then
205     begin
206         flag := true;
207         writeln('O Html contem um erro na linha ', errobody);
208     end;
209 end;
210
211 procedure verificaerroh1(pilhah1 : pilha; var flag : boolean; erroh1 :
        byte);
212 begin
213     if (not(vazia(pilhah1))) then
214     begin
215         flag := true;
216         writeln('O Html contem um erro na linha ', erroh1);
217     end;
218 end;
219
220 procedure verificaerrocenter(pilhacenter : pilha; var flag : boolean;
        errocenter : byte);
221 begin
222     if (not(vazia(pilhacenter))) then
223     begin
224         flag := true;
225         writeln('O Html contem um erro na linha ', errocenter);
226     end;
227 end;
228
```

```
229 procedure verificaerrool(pilhaol : pilha; var flag : boolean; errool :
        byte);
230 begin
231     if (not(vazia(pilhaol))) then
232     begin
233         flag := true;
234         writeln('O Html contem um erro na linha ', errool);
235     end;
236 end;
237
238 procedure verificaerroli(pilhali : pilha; var flag : boolean; erroli :
        byte);
239 begin
240     if (not(vazia(pilhali))) then
241     begin
242         flag := true;
243         writeln('O Html contem um erro na linha ', erroli);
244     end;
245 end;
246
247 var
248     pilhahtml, pilhahead, pilhatitle, pilhabody, pilhah1, pilhacenter,
            pilhaol, pilhali: pilha;
249     numlinha, errohtml, errohead, errotitle, errobody, erroh1,
            errocenter, errool, erroli : byte;
250     s : string;
251     f : text;
252     flag : boolean;
253
254 begin
255     clrscr;
256     numlinha := 0;
257     criar(pilhahtml);
258     criar(pilhahead);
259     criar(pilhatitle);
260     criar(pilhabody);
261     criar(pilhah1);
262     criar(pilhacenter);
263     criar(pilhaol);
264     criar(pilhali);
265     flag := false;
266
267     // Verificando se o arquivo html foi passado como argumento
268
269     if (ParamCount <> 1) then
270     begin
271         writeln('Numero errado de argumentos! Voce deve passar um
                arquivo html como parametro!');
272         exit;
273     end;
274
275     assign (f,ParamStr(1));
276      reset(f);
277
278     while (not eof(f)) do
279     begin
```

```
280         readln(f,s);
281         inc(numlinha); // contador pra saber em que linha tera algum
                erro
282         verificahtml(pilhahtml, s, numlinha, errohtml); // verifica tag
                <html>
283         verificahead(pilhahead, s, numlinha, errohead); // verifica tag
                <head>
284         verificatitle(pilhatitle, s, numlinha, errotitle); // verifica
                tag <title>
285         verificabody(pilhabody, s, numlinha, errobody); // verifica tag
                <body>
286         verificah1(pilhah1, s, numlinha, erroh1); // verifica tag <h1>
287         verificacenter(pilhacenter, s, numlinha, errocenter); //
                verifica tag <center>
288         verificaol(pilhaol, s, numlinha, errool); // verifica tag <ol>
289         verificali(pilhali, s, numlinha, erroli); // verifica tag <li>
290     end;
291     close(f);
292
293     verificaerrohtml(pilhahtml, flag, errohtml); // verifica se ha erro
            na tag <html>
294     verificaerrohead(pilhahead, flag, errohead); // verifica se ha erro
            na tag <head>
295     verificaerrotitle(pilhatitle, flag, errotitle); // verifica se ha
            erro na tag <title>
296     verificaerrobody(pilhabody, flag, errobody); // verifica se ha erro
            na tag <body>
297     verificaerrohead(pilhah1, flag, erroh1); // verifica se ha erro na
            tag <h1>
298     verificaerrocenter(pilhacenter, flag, errocenter); // verifica se
            ha erro na tag <center>
299     verificaerrool(pilhaol, flag, errool); // verifica se ha erro na
            tag <ol>
300     verificaerroli(pilhali, flag, erroli); // verifica se ha erro na
            tag <li>
301
302     if not(flag) then // Se a flag nao subiu significa que nao ha erro
            no html
303         writeln('Arquivo HTML esta correto!');
304
305     readln;
306 end.
```

corpo.pas

5.2. Biblioteca

```
1  {
2      Bruno Tome - 0011254 - ibrunotome@gmail.com
3      Claudio Menezes - 0011255 - claudiomenezio.@gmail.com
4  }
5
6  unit biblioteca;
7
8  interface
9
10 type
```

```
11      elem = byte;
12
13      noh = record
14          item: elem;
15          proximo: ^noh;
16      end;
17
18      pilha = record
19            topo: ^noh;
20      end;
21
22  procedure criar(var p:pilha);
23  function vazia (var p:pilha):boolean;
24  function inserir (var p:pilha; x:elem): boolean;
25  function remover (var p:pilha): boolean;
26  function topo(var p:pilha): elem;
27
28
29  implementation
30
31  procedure criar(var p:pilha);
32      begin
33          p.topo:=nil;
34      end;
35
36  function vazia (var p:pilha):boolean;
37      begin
38          if p.topo=nil then
39              vazia:=true
40          else
41              vazia:=false;
42      end;
43
44  function inserir (var p:pilha; x:elem): boolean;
45      var
46          novo_noh: ^noh;
47      begin
48          new(novo_noh); { criar novo noh }
49          if (novo_noh <> nil) then
50            begin
51               novo_noh^.item := x;
52                 novo_noh^.proximo := p.topo;
53               p.topo := novo_noh;
54                 inserir:=true;
55            end
56          else
57              inserir:=false;
58      end;
59
60
61  function remover (var p:pilha): boolean;
62      var
63          aux : ^noh; { armazena o noh a ser excluido }
64      begin
65          if not vazia(p) then
66          begin
```

```
67              aux := p.topo;
68              p.topo := p.topo^.proximo;
69              dispose (aux);
70              remover := true;
71          end
72          else
73              remover:=false;
74      end;
75
76
77  function topo(var p:pilha): elem;
78      begin
79          topo:=p.topo^.item;
80      end;
81  end.
```

biblioteca.pas