

Estrutura de Dados – Projeto I

Universidade Federal de Santa Catarina

Matheus Aparicio da Silva

20100538

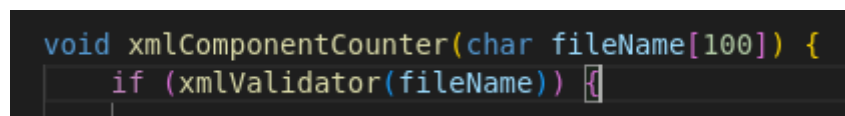
O projeto, que consiste em fazer análises de arquivos XML foi concluído parcialmente.

Nele é possível apenas validar um arquivo XML, a parte responsável por fazer a análise das imagens binárias não foi implementada.



```
Project I > main.cpp > main()
1  #include <iostream>
2  #include "xml_component_counter.h"
3
4  int main() {
5
6      char fileName[100];
7
8      std::cin >> fileName; // entrada
9
10     xmlComponentCounter(fileName);
11
12     return 0;
13 }
```

Na função principal do programa, o xmlComponentCounter é chamado de acordo com o input do usuário.



```
void xmlComponentCounter(char fileName[100]) {
    if (xmlValidator(fileName)) {
```

Caso o arquivo XML seja válido, a lógica (ainda incompleta) do xmlComponentCounter é aplicada. Caso o XML não seja válido, é printado “error” na tela.

```

33 //Função para validar arquivos XML.
34 bool xmlValidator(char fileName[100]) {
35     // Variáveis declaração de algumas variáveis.
36     int lineCount = 0;
37     structures::ArrayStack<string> arrayStack(10);
38     string line;
39     ifstream in(fileName);
40     while (getline(in, line)) {
41         for (int i = 0; i < line.size(); i++) {
42             if (line[i] == '<') { // Se encontrar um '<'.
43                 if (line[i+1] != '/') { // Se for uma tag de abertura.
44                     arrayStack.push(getTag(line.substr(i)));
45                 } else { // Se for uma tag de encerramento.
46                     if (arrayStack.size() < 1 || convertClosingToOpeningTag(getTag(line.substr(i))) != arrayStack.top()) { //
47                         return 0;
48                     } else {
49                         arrayStack.pop();
50                     }
51                 }
52             }
53         }
54         lineCount++;
55     }
56     if (arrayStack.size() > 0) {
57         return 0;
58     }
59     return 1;
60 }

```

A lógica do validador consiste em procurar por ‘<’ no código e, ao achar, caso seja uma tag de abertura, ela é inserida na pilha e, caso seja uma tag de encerramento, é feita a verificação de erros e, caso não haja erro, é removido um elemento da pilha.

```

25 // Função para converter uma tag de encerramento para uma de abertura.
26 string convertClosingToOpeningTag(string closingTag) {
27
28     string convertedTag = '<' + closingTag.substr(2);
29     return convertedTag;
30 }
31 }

```

A validação foi feita usando duas funções auxiliares. ConvertClosingToOpeningTag, do qual o nome já indica sua finalidade.

```

8 // Função feita para pegar a tag completa quando um '<' for encontrado.
9 string getTag(string line) {
10
11     string tag = "";
12     for (int i = 0; i < line.length(); i++) {
13         if (line[i] != '>') {
14             tag += line[i];
15         } else {
16             tag += line[i];
17             break;
18         }
19     }
20
21     return tag;
22 }
23 }

```

E getTag, responsável por retornar a tag completa quando um ‘<’ for achado no XML.

Foi um trabalho divertido de se fazer e apenas não o concluí por má gestão de tempo. Não tive muitas dificuldades específicas além da lógica do analisador de imagens binárias. No geral foi possível sanar minhas dúvidas com rápidas pesquisas na internet.

```
        break;
    }

    int valueInt = atoi(valueString.c_str());

    return valueInt;
```

Creio que, fora da lógica de implementação, o que mais me consumiu tempo foi a pesquisa de como converter uma string para um int.

Alguns links que me ajudaram bastante:

<https://stackoverflow.com/questions/5443073/read-a-line-from-xml-file-using-c>

<https://www.geeksforgeeks.org/multidimensional-arrays-c-cpp/>

<https://stackoverflow.com/questions/7663709/how-can-i-convert-a-stdstring-to-int>

<https://www.geeksforgeeks.org/converting-strings-numbers-c-cpp/>