Projeto Estrutura de Dados

XML Parser

Thiago Z L Chaves (19100547)
Samuel Cardoso (19100544)
Professor: Alexandre e Aldo
Quinta, 15 de Outubro de 2020

Sumário

Índice dos Arquivos

Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

Arquivos

Referência do Arquivo recursiveXML.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
```

Funções

- int **recursive** (string *content, int *height, int *width, int j, int k)
  recursive

- int **recursiveXML** (string *content, int *height, int *width)

Funções

int recursive (string * *content*, int * *height*, int * *width*, int *j*, int *k*)

O método faz uso de recursão para percorrer um vetor, este que tem papel de simular uma matriz. Durante o loop, o principal objetivo é encontrar campos adjacentes onde o valor em sua posição seja ‘1’, por conseguinte, quando encontros, a posição deste novo elemento é definida como a nova posição a ser analisada, seu valor é alterado para 2, para que a recursão não acabe pegando o valor repetidamente. Então a análise do próximo elemento é iniciada e processo se repete até que não sejam mais encontrados campos com o valor ‘1’ em volta. **int**

recursiveXML (string * *content*, int * *height*, int * *width*)

O âmago desta função é o mesmo da função que a antecede, se tem como base um vetor, e este é usado como se fosse uma matriz. Porém agora o fato a se consumir, é estabelecer quantos objetos existem no XML, e por tanto é aqui que esta informação é guardada. Enquanto o vetor é percorrido, cada vez que o valor do elemento for ‘1’, o valor total de elementos recebe um acréscimo de 1 e a partir daquele ponto inicia um processo de recursão bloqueando todos os valores ‘1’ que estão ao redor, para que o elemento não seja contato de forma errônea.

```
int recursive(string *content, int *height, int *width, int j, int k) {

    //! Seta o valor atual para 2 para que não seja repetido seu valor
    if ((*content)[j*(*width)+k] == '1') {
        (*content)[j*(*width)+k] = '2';
    }

    //! Bloquei a linha de cima
    if ((j*(*width)+k) > ((*width)-1)) {
        //! Percorre para cima
        if ((*content)[(j-1)*(*width)+k] == '1') {
            recursive(content, height, width, j-1, k);
        }
    }
```

Referência do Arquivo XMLValidation.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include "array_stack.h"
#include "recursiveXML.h"
```

Funções

- bool **validation** (char arquivo[100])

Funções

bool validation (char *arquivo*[100])

Função serve para validação dos arquivos XML e extrair seus dados:
-> Imagens
-> Dimensões
É recebido o XML como parametro da função e então o programa inicia sua execução realizando os passos a seguir:
1) Abre o arquivo XML.
2) Realiza a verificação para confirmar que o arquivo foi aberto com sucessor.
3) Percorre as linhas procurando um "<". Quando encontra, define a variável onTag como 'True', está variável que é responsável por quando os elementos a seguir devem ser adicionados na String tag. Além disso, é ao passar pelo "<" que a variável onContent é setada com "False" e os conteúdo da String content é resetada.
4) Por tant Quando o cursor está na tag a String tag é escrita.
5) Quando o cursor está nos dados a String contents é escrita.
6) Quando verifica que existe um '/', o programa entende que é a tag atual é uma tag de fechamento.
7) Durante o Loop do programa, ao encontrarmos um ">" o programa realização inversa da operação 3, setando agora onContent como "True", e assim passando a adicionar caracteres na String content, e setando a variável onTag como "False". Caso a tag atual seja de fechamento, é feito uma comparação entre a tag e o topo da nossa pilha, caso eles sejam iguais é realizada uma operação com base na resposta, caso não é emitido um erro.
8) Conforme o programa vai executando, ele vai comparando as tags e tomando ações conforme as tag atuais, quando a tag é "height", a altura da imagem é guardada no variável int height, quando a tag é "width", a largura da imagem é guardada na variável int width, quando a tag é "name", o nome da imagem é guardado na variável String name e por último, quando a tag é "data", a função recursiveXML é chamada, levando consigo todos os dados acima sitados mais a String content, que no atual momento guarda a imagem em si.
9) O programa executa, e percorre todas as imagens do arquivo XML, salvando o nome da imagem e quantidade de elementos dela na String result que será mandada como resposta no final do programa.
10) Verifica se todos os elementos da pilha foram retirados caso contrário é retornado um erro.
11) Retorna o valor esperado pelo Projeto XML Parser.

Referência do Arquivo main.cpp

```
#include <iostream>
#include <fstream>
#include "XMLValidation.h"
```

Funções

- int main ()

Funções

int main( )

Funcão criar variavel xmlfilename.

Recebe o XML do terminal e é alocado na variável char xmlfilename.

Chama a função validation.

```
#include <iostream>
#include <fstream>
#include "XMLValidation.h"

using namespace std;

int main() {

    char xmlfilename[100];

    cin >> xmlfilename;  // Entra (Nome do arquivo)

    validation(xmlfilename); // Função responsável por validar e iniciar a segunda etápa

    return 0;
}
```

Dificuldades Apresentadas

Nesse trabalho obtivemos 2 principais complicações durante seu desenvolvimento: o primeiro, devido a nossa falta de experiência com C++ e o não conhecimento de certas bibliotecas, acabamos pendurando muito para realizar o Parse de XML em si, colocando seu conteúdo em uma string, já a segunda complicação foi em passar parâmetros de uma função para outra, misturando ponteiros, endereços e os conteúdos em si, além das diferenças de tipo, coisa que agora estamos aprendendo já que viemos de linguagens como Python e JavaScript, onde a tipagem não é forte.