Projeto I - Processamento de XML com imagens binárias

Gerado por Doxygen 1.8.13

Sumário

Capítulo 1

Namespaces

1.1 Lista de Namespaces

Esta é a lista de todos os Namespaces com suas respectivas descrições:

Capítulo 2

Índice dos Componentes

2.1 Lista de Componentes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Capítulo 3

Índice dos Arquivos

3.1 Lista de Arquivos

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

Capítulo 4

Namespaces

4.1 Refência do Namespace structures

Estruturas de dados.

Componentes

- class LinkedQueue

  Fila Encadeada.
- class LinkedStack

  Pilha Encadeada.

4.1.1 Descrição Detalhada

Estruturas de dados.

4.2 Refência do Namespace xml_processing

Código para processamento de xml.

Componentes

- class Matrix

  Matriz.
- class XML

  Xml.

4.2.1 Descrição Detalhada

Código para processamento de xml.

Capítulo 5

Classes

5.1 Referência da Template de Classe structures::LinkedQueue< T >

Fila Encadeada.

`#include <linked_queue.h>`

Métodos Públicos

- LinkedQueue ()
  Construtor padrão.
- ~LinkedQueue ()
  Destrutor.
- void clear ()
  Limpa fila.
- void enqueue (const T &data)
  Enfilerar.
- T dequeue ()
  Desenfilerar.
- T & front () const
  Primeiro dado da fila.
- T & back () const
  Último dado da fila.
- bool empty () const
  Verifica se a fila está vazia.
- std::size_t size () const
  Tamanho da fila.

5.1.1 Descrição Detalhada

template<typename T>
class structures::LinkedQueue< T >

Fila Encadeada.

5.1.2 Construtores & Destrutores

5.1.2.1 LinkedQueue()

```
template<typename T >
structures::LinkedQueue< T >::LinkedQueue ( )
```

Construtor padrão.

5.1.2.2 ~LinkedQueue()

```
template<typename T >
structures::LinkedQueue< T >::~LinkedQueue ( )
```

Destrutor.

5.1.3 Métodos

5.1.3.1 back()

```
template<typename T >
T & structures::LinkedQueue< T >::back ( ) const
```

Último dado da fila.

5.1.3.2 clear()

```
template<typename T >
void structures::LinkedQueue< T >::clear ( )
```

Limpa fila.

5.1.3.3 dequeue()

```
template<typename T >
T structures::LinkedQueue< T >::dequeue ( )
```

Desenfilerar.

5.1.3.4 empty()

```
template<typename T >
bool structures::LinkedQueue< T >::empty ( ) const
```

Verifica se a fila está vazia.

5.1.3.5 enqueue()

```
template<typename T >
void structures::LinkedQueue< T >::enqueue (
            const T & data )
```

Enfilerar.

5.1.3.6 front()

```
template<typename T >
T & structures::LinkedQueue< T >::front ( ) const
```

Primeiro dado da fila.

5.1.3.7 size()

```
template<typename T >
std::size_t structures::LinkedQueue< T >::size ( ) const
```

Tamanho da fila.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/linked_queue.h

5.2 Referência da Template de Classe structures::LinkedStack< T >

Pilha Encadeada.

```
#include <linked_stack.h>
```

Métodos Públicos

- LinkedStack ()
  Construtor padrão.
- ~LinkedStack ()
  Destrutor.
- void clear ()
  Limpar pilha.
- void push (const T &data)
  Empilha.
- T pop ()
  Desempilha.
- T & top () const
  Topo da pilha.
- bool empty () const
  Verifica se a pilha está vazia.
- std::size_t size () const
  Tamanho da pilha.

5.2.1 Descrição Detalhada

template<typename T>
class structures::LinkedStack< T >

Pilha Encadeada.

5.2.2 Construtores & Destrutores

5.2.2.1 LinkedStack()

```
template<typename T >
structures::LinkedStack< T >::LinkedStack ( )
```

Construtor padrão.

5.2.2.2 ~LinkedStack()

```
template<typename T >
structures::LinkedStack< T >::~LinkedStack ( )
```

Destrutor.

5.2.3 Métodos

5.2.3.1 clear()

```
template<typename T >
void structures::LinkedStack< T >::clear ( )
```

Limpar pilha.

5.2.3.2 empty()

```
template<typename T >
bool structures::LinkedStack< T >::empty ( ) const
```

Verifica se a pilha está vazia.

5.2.3.3 pop()

```
template<typename T >
T structures::LinkedStack< T >::pop ( )
```

Desempilha.

5.2.3.4 push()

```
template<typename T >
void structures::LinkedStack< T >::push (
            const T & data )
```

Empilha.

5.2.3.5 size()

```
template<typename T >
std::size_t structures::LinkedStack< T >::size ( ) const
```

Tamanho da pilha.

5.2.3.6 top()

```
template<typename T >
T & structures::LinkedStack< T >::top ( ) const
```

Topo da pilha.

A documentação para esta classe foi gerada a partir do seguinte arquivo:

- include/linked_stack.h

5.3 Referência da Classe xml_processing::Matrix

Matriz.

`#include <matrix.h>`

Métodos Públicos

- Matrix (int n_lines, int n_columns)
  Constroi uma matriz.
- ~Matrix ()
  Destrói a matriz.
- int count_connected ()
  Conta componentes conexos na matriz.
- int get_value (int line, int column)
  Pega uma posição da matriz.
- void set_value (int line, int column, int value)
  Coloca um inteiro na posição passada.

5.3.1 Descrição Detalhada

Matriz.

5.3.2 Construtores & Destrutores

5.3.2.1 Matrix()

```
xml_processing::Matrix::Matrix (
            int n_lines,
            int n_columns )
```

Constroi uma matriz.

Constroi uma matriz do tipo ponteiro para ponteiro de inteiros. Esta matriz será nula, ou seja, todos os elementos são zero.

Parâmetros

n_lines	Número de linhas da matriz.
n_columns	Número de colunas da matriz.

5.3.2.2 ∼Matrix()

```
xml_processing::Matrix::∼Matrix ( )
```

Destrói a matriz.

Libera todo o espaço de memória ocupado pela matriz.

5.3.3 Métodos

5.3.3.1 count_connected()

```
int xml_processing::Matrix::count_connected ( )
```

Conta componentes conexos na matriz.

A matriz neste caso é uma imagem binária. Cada pixel conexo encontrado na matriz é rotulado com um valor inteiro e enfileirado utilizando uma fila encadeada. Utilizamos vizinhança-4 para o calculo de componentes conexos. A vizinhança-4 de um pixel são os pixels que estão em cima, em baixo, na direita e na esquerda.

Retorna

Inteiro com o número de componentes conexos na matriz.

5.3.3.2 get_value()

```
int xml_processing::Matrix::get_value (
            int line,
            int column )
```

Pega uma posição da matriz.

Parâmetros

line	Linha onde será definido o valor.
column	Coluna onde será definido o valor.

Retorna

Inteiro que está na posição passada.

5.3.3.3 set_value()

```
void xml_processing::Matrix::set_value (
            int line,
            int column,
            int value )
```

Coloca um inteiro na posição passada.

Parâmetros

line	Linha onde será definido o valor.
column	Coluna onde será definido o valor.
value	Valor que será colocado na matriz.

A documentação para esta classe foi gerada a partir dos seguintes arquivos:

- include/matrix.h
- src/matrix.cpp

5.4 Referência da Classe xml_processing::XML

Xml.

```
#include <xml.h>
```

Métodos Públicos

- XML (const std::string &xml_string)

  Construtor.
- bool validation ()

  Faz a validação do arquivo xml passado.
- std::string extract (const std::string &tag_begin, const std::string &tag_end, std::size_t &position)

  Faz a extração do conteúdo entre as tags.

5.4.1 Descrição Detalhada

Xml.

5.4.2 Construtores & Destrutores

5.4.2.1 XML()

```
xml_processing::XML::XML (
            const std::string & xml_string )
```

Construtor.

Recebe o arquivo xml no formado string para ser utilizado no restante dos metodos.

Parâmetros

xml	Arquivo xml no formado string.

5.4.3 Métodos

5.4.3.1 extract()

```
std::string xml_processing::XML::extract (
            const std::string & tag_begin,
            const std::string & tag_end,
            std::size_t & position )
```

Faz a extração do conteúdo entre as tags.

A extração copia o conteúdo entre as tags que estão a partir da posição passada para iniciar a busca no xml em formato de string.

Parâmetros

tag_begin	Indicador de abertura de uma tag.
tag_end	Indicador de fechamento de uma tag.
position	Posição inicial de busca no xml em formato string.

Retorna

String com o conteúdo entre as tags.

5.4.3.2 validation()

```
bool xml_processing::XML::validation ( )
```

Faz a validação do arquivo xml passado.

A validação verifica se todas as tags abriram e fecharam corretamente. É utilizado uma pilha encadeada auxiliar neste processo.

Retorna

True se o arquivo for válido e false se for inválido.

A documentação para esta classe foi gerada a partir dos seguintes arquivos:

- include/xml.h
- src/xml.cpp

Capítulo 6

Arquivos

6.1 Referência do Arquivo include/linked_queue.h

Código da fila encadeada.

```
#include <cstdint>
#include <stdexcept>
```

Componentes

- class structures::LinkedQueue< T >

  Fila Encadeada.

Namespaces

- structures

  Estruturas de dados.

6.1.1 Descrição Detalhada

Código da fila encadeada.

Autores

Alisson Fabra da Silva e Eduardo Vinicius Betim.

Copyright

6.2 Referência do Arquivo include/linked_stack.h

Código da pilha encadeada.

```
#include <cstdint>
#include <stdexcept>
```

Componentes

- class structures::LinkedStack< T >

  Pilha Encadeada.

Namespaces

- structures

  Estruturas de dados.

6.2.1 Descrição Detalhada

Código da pilha encadeada.

Autores

Alisson Fabra da Silva e Eduardo Vinicius Betim.

Copyright

Copyright [2021] <Alisson Fabra da Silva, Eduardo Vinicius Betim>

6.3 Referência do Arquivo include/matrix.h

Declaração da classe Matrix.

Componentes

- class xml_processing::Matrix

  Matriz.

Namespaces

- xml_processing

  Código para processamento de xml.

6.3.1 Descrição Detalhada

Declaração da classe Matrix.

Autores

Alisson Fabra da Silva e Eduardo Vinicius Betim.

Copyright

6.4 Referência do Arquivo include/xml.h

Declaração das funções validação e extração de conteúdo de arquivos xml.

```
#include <string>
#include <cstddef>
```

Componentes

- class xml_processing::XML

  Xml.

Namespaces

- xml_processing

  Código para processamento de xml.

6.4.1 Descrição Detalhada

Declaração das funções validação e extração de conteúdo de arquivos xml.

Autores

Alisson Fabra da Silva e Eduardo Vinicius Betim.

Copyright

6.5 Referência do Arquivo main.cpp

Código principal.

```
#include <iostream>
#include <fstream>
#include <string>
#include <cctype>
#include "./include/xml.h"
#include "./include/matrix.h"
```

Funções

- int main ()

  Função main, ponto de partida para a execução do programa.

6.5.1 Descrição Detalhada

Código principal.

Autores

Alisson Fabra da Silva e Eduardo Vinicius Betim.

Copyright

Copyright [2021] <Alisson Fabra da Silva, Eduardo Vinicius Betim>

6.5.2 Funções

6.5.2.1 main()

```
int main ( )
```

Função main, ponto de partida para a execução do programa.

Recebe um arquivo xml com uma imagem binária no input, valida o arquivo e caso seja válido processa o imagem.

Retorna

Retona 0 caso não ocorra erro, -1 caso o xml não abra, -2 caso o xml seja inválido e -3 caso a imagem do xml tenha dimensão inválida.

6.6 Referência do Arquivo src/matrix.cpp

Implementação da classe Matrix.

```
#include <utility>
#include "../include/linked_queue.h"
#include "../include/matrix.h"
```

6.6.1 Descrição Detalhada

Implementação da classe Matrix.

Autores

Alisson Fabra da Silva e Eduardo Vinicius Betim.

Copyright

6.7 Referência do Arquivo src/xml.cpp

Implementação das funções validação e extração de arquivos XML.

```
#include <string>
#include "../include/linked_stack.h"
#include "../include/xml.h"
```

6.7.1 Descrição Detalhada

Implementação das funções validação e extração de arquivos XML.

Autores

Alisson Fabra da Silva e Eduardo Vinicius Betim.

Copyright

Índice Remissivo