xml-shapes

v0

Gerado por Doxygen 1.8.15

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 Classes

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

Namespace

4.1 Refência do Namespace structures

Componentes

- class LinkedQueue

  Fila Encadeada.
- class LinkedStack

  Pilha Encadeada.

4.2 Refência do Namespace xml

Funções

- bool balanced (const std::string &xml)

  Confere a validez da estrutura do XML contido na string.
- std::string extract (const std::string &origin, const std::string &open, const std::string &close, std::size_t &from)

  Extrai substring de uma string entre dois delimitadores encontrados a partir de uma dada posicao.
- std::string extract (const std::string &origin, const std::string &open, const std::string &close)

  Extrai substring de uma string entre dois delimitadores.

4.2.1 Funções

4.2.1.1 balanced()

```
bool xml::balanced (
            const std::string & xml )
```

Confere a validez da estrutura do XML contido na string.

Definição na linha 12 do arquivo xml.cpp.

Referenciado(a) por main().

4.2.1.2 extract() [1/2]

```
std::string xml::extract (
             const std::string & origin,
             const std::string & open,
             const std::string & close,
             std::size_t & from )
```

Extrai substring de uma string entre dois delimitadores encontrados a partir de uma dada posicao.

Retorna a substring extraida (sem delimitadores) e altera a posicao para o final do ultimo delimitador encontrado. Se nao encontrar nada, retorna string vazia e leva a posicao para npos.

Definição na linha 53 do arquivo xml.cpp.

Referenciado(a) por extract() e main().

4.2.1.3 extract() [2/2]

```
std::string xml::extract (
             const std::string & origin,
             const std::string & open,
             const std::string & close )
```

Extrai substring de uma string entre dois delimitadores.

Retorna a substring extraida (sem delimitadores). Se nao encontrar nada, retorna string vazia.

Definição na linha 70 do arquivo xml.cpp.

Capítulo 5

Classes

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

Fila Encadeada.

`#include <linked_queue.h>`

Membros Públicos

- ~LinkedQueue ()
  Destrutor.
- void clear ()
  Limpa a Fila.
- void enqueue (const T &data)
  Enfileira.
- T dequeue ()
  Desenfileira.
- T & front () const
  Acessa a frente da Fila.
- T & back () const
  Acessa o ultimo da Fila.
- bool empty () const
  Confere se a Fila esta vazia.
- std::size_t size () const
  Retorna o tamanho da Fila.

5.1.1 Descrição detalhada

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

Fila Encadeada.

Definição na linha 12 do arquivo linked_queue.h.

5.1.2 Construtores e Destrutores

5.1.2.1 ~LinkedQueue()

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

Destrutor.

Definição na linha 5 do arquivo linked_queue.h.

5.1.3 Funções membros

5.1.3.1 back()

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

Acessa o ultimo da Fila.

Definição na linha 59 do arquivo linked_queue.h.

5.1.3.2 clear()

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

Limpa a Fila.

Definição na linha 10 do arquivo linked_queue.h.

5.1.3.3 dequeue()

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

Desenfileira.

Definição na linha 25 do arquivo linked_queue.h.

Referenciado(a) por count_shapes().

5.1.3.4 empty()

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

Confere se a Fila esta vazia.

Definição na linha 67 do arquivo linked_queue.h.

Referenciado(a) por count_shapes().

5.1.3.5 enqueue()

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

Enfileira.

Definição na linha 16 do arquivo linked_queue.h.

Referenciado(a) por count_shapes().

5.1.3.6 front()

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

Acessa a frente da Fila.

Definição na linha 51 do arquivo linked_queue.h.

5.1.3.7 size()

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

Retorna o tamanho da Fila.

Definição na linha 72 do arquivo linked_queue.h.

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

- src/linked_queue.h
- src/linked_queue.inc

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

Pilha Encadeada.

`#include <linked_stack.h>`

Membros Públicos

- ~LinkedStack ()
  Destrutor.
- void push (const T &data)
  Empilha.
- T pop ()
  Desempilha.
- T & top () const
  Acessa o topo da Pilha.
- bool empty () const
  Confere se a Pilha esta vazia.
- std::size_t size () const
  Retorna o tamanho da Pilha.
- void clear ()
  Limpa a Pilha.

5.2.1 Descrição detalhada

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

Pilha Encadeada.

Definição na linha 12 do arquivo linked_stack.h.

5.2.2 Construtores e Destrutores

5.2.2.1 ~LinkedStack()

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

Destrutor.

Definição na linha 5 do arquivo linked_stack.h.

5.2.3 Funções membros

5.2.3.1 clear()

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

Limpa a Pilha.

Definição na linha 10 do arquivo linked_stack.h.

5.2.3.2 empty()

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

Confere se a Pilha esta vazia.

Definição na linha 49 do arquivo linked_stack.h.

Referenciado(a) por xml::balanced().

5.2.3.3 pop()

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

Desempilha.

Definição na linha 22 do arquivo linked_stack.h.

Referenciado(a) por xml::balanced().

5.2.3.4 push()

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

Empilha.

Definição na linha 16 do arquivo linked_stack.h.

Referenciado(a) por xml::balanced().

5.2.3.5 size()

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

Retorna o tamanho da Pilha.

Definição na linha 54 do arquivo linked_stack.h.

5.2.3.6 top()

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

Acessa o topo da Pilha.

Definição na linha 41 do arquivo linked_stack.h.

Referenciado(a) por xml::balanced().

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

- src/linked_stack.h
- src/linked_stack.inc

Capítulo 6

Arquivos

6.1 Referência do Arquivo src/linked_queue.h

```
#include <cstdint>
#include <stdexcept>
#include "linked_queue.inc"
```

Componentes

- class structures::LinkedQueue< T >

  Fila Encadeada.

Namespaces

- structures

6.2 Referência do Arquivo src/linked_queue.inc

6.3 Referência do Arquivo src/linked_stack.h

```
#include <cstdint>
#include <stdexcept>
#include "linked_stack.inc"
```

Componentes

- class structures::LinkedStack< T >

  Pilha Encadeada.

Namespaces

- structures

6.4 Referência do Arquivo src/linked_stack.inc

6.5 Referência do Arquivo src/main.cpp

```
#include <iostream>
#include <string>
#include <fstream>
#include <sstream>
#include <cctype>
#include <cassert>
#include <utility>
#include "xml.h"
#include "linked_queue.h"
```

Funções

- static int ** matrix_init (int height, int width)

  Inicializa uma matriz como um array de height ponteiros para arrays com width ints. Deve ser destruido com matrix_destroy() para liberar a memoria.
- static int ** matrix_init (int height, int width, const std::string &data)

  Inicializa uma matriz a partir da string que representa seus valores. Deve ser destruido com matrix_destroy() para liberar a memoria.
- static void matrix_destroy (int **M, int height)

  Utilizado para liberar a memoria alocada por matrix_init().
- static int count_shapes (int **E, int height, int width)

  Calcula o numero de componentes conexos na matriz usando vizinhanca-4.
- int main ()

6.5.1 Funções

6.5.1.1 count_shapes()

```
static int count_shapes (
            int ** E,
            int height,
            int width )  [static]
```

Calcula o numero de componentes conexos na matriz usando vizinhanca-4.

Definição na linha 80 do arquivo main.cpp.

Referenciado(a) por main().

6.5.1.2 main()

```
int main ( )
```

Definição na linha 30 do arquivo main.cpp.

6.5.1.3 matrix_destroy()

```
static void matrix_destroy (
            int ** M,
            int height )  [static]
```

Utilizado para liberar a memoria alocada por matrix_init().

Definição na linha 143 do arquivo main.cpp.

Referenciado(a) por count_shapes() e main().

6.5.1.4 matrix_init() [1/2]

```
static int ** matrix_init (
            int height,
            int width )  [static]
```

Inicializa uma matriz como um array de height ponteiros para arrays com width ints. Deve ser destruido com matrix_destroy() para liberar a memoria.

Definição na linha 131 do arquivo main.cpp.

Referenciado(a) por count_shapes(), main() e matrix_init().

6.5.1.5 matrix_init() [2/2]

```
static int ** matrix_init (
            int height,
            int width,
            const std::string & data )  [static]
```

Inicializa uma matriz a partir da string que representa seus valores. Deve ser destruido com matrix_destroy() para liberar a memoria.

Definição na linha 149 do arquivo main.cpp.

6.6 Referência do Arquivo src/xml.cpp

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

Namespaces

- xml

Funções

- bool xml::balanced (const std::string &xml)

  Confere a validez da estrutura do XML contido na string.
- std::string xml::extract (const std::string &origin, const std::string &open, const std::string &close, std::size_t &from)

  Extrai substring de uma string entre dois delimitadores encontrados a partir de uma dada posicao.
- std::string xml::extract (const std::string &origin, const std::string &open, const std::string &close)

  Extrai substring de uma string entre dois delimitadores.

6.7 Referência do Arquivo src/xml.h

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

Namespaces

- xml

Funções

- bool xml::balanced (const std::string &xml)

  Confere a validez da estrutura do XML contido na string.
- std::string xml::extract (const std::string &origin, const std::string &open, const std::string &close, std::size_t &from)

  Extrai substring de uma string entre dois delimitadores encontrados a partir de uma dada posicao.
- std::string xml::extract (const std::string &origin, const std::string &open, const std::string &close)

  Extrai substring de uma string entre dois delimitadores.

Índice Remissivo