Trabalho de Implementação I

Gilson Trombetta Magro

Generated by Doxygen 1.8.13

Contents

1 Namespace Index **1**

1.1 Namespace List 1

2 Class Index **3**

2.1 Class List 3

3 Namespace Documentation **5**

3.1 functions Namespace Reference 5

3.1.1 Detailed Description 5

3.1.2 Function Documentation 5

3.1.2.1 build_matrix() 5

3.1.2.2 connected_components() 6

3.1.2.3 process_xml_file() 6

3.1.2.4 verify_xml_file() 7

3.2 structures Namespace Reference 7

3.2.1 Detailed Description 8

4 Class Documentation **9**

4.1 structures::LinkedQueue< T > Class Template Reference 9

4.1.1 Detailed Description 9

4.2 structures::LinkedStack< T > Class Template Reference 10

4.2.1 Detailed Description 10

4.3 structures::pos_s Struct Reference 10

4.3.1 Detailed Description 10

Index **11**

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

functions
Namespace que engloba as funções do trabalho 5
structures
Namespace que engloba as estruturas utilizadas 7

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

structures::LinkedQueue< T >
: Classe fila encadeada (dinâmica) 9

structures::LinkedStack< T >
: Classe pilha encadeada (dinâmica) 10

structures::pos_s
: Struct que representa a posição (x, y) de um píxel na imagem 10

Chapter 3

Namespace Documentation

3.1 functions Namespace Reference

Namespace que engloba as funções do trabalho.

Functions

- bool verify_xml_file (const string filename)

  Função que valida um arquivo XML.
- int * build_matrix (int width, int height)

  Função que constroi uma matriz representada em um vetor e preenche-a com zeros.
- int connected_components (const string name, const string data, int width, int height)

  Função que encontra os elementos 4-conectados em uma imagem binária.
- void process_xml_file (const string filename)

  Função que itera pelo arquivo XML e processa todas as imagens binárias encontradas.

3.1.1 Detailed Description

Namespace que engloba as funções do trabalho.

O namespace "functions" contém as funções verify_xml_file, connected_components, build_matrix e process_↩ xml_file.

3.1.2 Function Documentation

3.1.2.1 build_matrix()

```
int* functions::build_matrix (
            int width,
            int height )
```

Função que constroi uma matriz representada em um vetor e preenche-a com zeros.

Parameters

width	um inteiro que corresponde à largura da matriz.
height	um inteiro que corresponde à altura da matriz.

Returns

um ponteiro de inteiro que aponta para a matriz criada.

A função build_matrix é usada para montar uma matriz de zeros, utilizada como matriz de resultado para o algoritmo de rotulação dos componentes conexos nas imagens binárias do XML. A matriz criada por essa função é na verdade representada por um vetor, por maior simplicidade na manipulação dos ponteiros. Essa matriz-vetor é alocada, preenchida com zeros e um ponteiro é retornado.

3.1.2.2 connected_components()

```
int functions::connected_components (
            const string name,
            const string data,
            int width,
            int height )
```

Função que encontra os elementos 4-conectados em uma imagem binária.

Parameters

name	uma string que representa o nome da imagem binária.
data	uma string contendo a imagem binária.
width	um inteiro que representa a largura da imagem.
height	um inteiro que representa a altura da imagem.

Returns

um inteiro que corresponde ao número de componentes 4-conectados na imagem.

A função connected_components recebe uma string "data" que representa uma imagem binária do XML. Ela recebe também o nome, a largura e a altura dessa imagem, e utiliza isso para encontrar e classificar cada componente conexo na imagem. São usadas funções lambda dentro da função para reduzir a repetição de código. O algoritmo de rotulação utilizado baseia-se em percorrer a imagem, e para cada valor 1 ainda não visitado na imagem, atribui-se o rótulo atual àquela posição, que depois é colocada na fila. Em seguida, enquanto a fila não estiver vazia, remove-se uma posição da fila e checa-se cada um de seus vizinhos (apenas aqueles que são posições válidas na imagem). Caso algum vizinho seja igual a 1 e ainda não tenha sido visitado, então este também recebe o rótulo atual e é colocado na fila. Quando a fila fica vazia, o rótulo atual é incrementado e o processo se repete até que todas as posições tenham sido visitadas. Ao final, retorna-se o valor do último rótulo atribuido menos 1.

3.1.2.3 process_xml_file()

```
void functions::process_xml_file (
            const string filename )
```

Função que itera pelo arquivo XML e processa todas as imagens binárias encontradas.

Parameters

filename	uma string que representa o nome do arquivo XML.

A função process_xml_file simplesmente itera pelo arquivo XML passado por parâmetro, caracter por caracter, da mesma forma que a função verify_xml_file, montando as tags que encontra. Quando encontra uma tag de fechamento "</data>", presume-se que já se tenha passado pelas tags "name", "width", "height" e a própria tag "data". Portanto, tem-se uma imagem binária com todos os seus atributos, e então se pode processar essa imagem utilizando a função connected_components.

3.1.2.4 verify_xml_file()

```
bool functions::verify_xml_file (
            const string filename )
```

Função que valida um arquivo XML.

Parameters

filename	uma string que representa o caminho do arquivo XML.

Returns

um boolean.

A função verify_xml_file itera sobre o arquivo XML, cujo caminho é passado como parâmetro, e valida o posicionamento das tags. Ela itera por cada caracter do arquivo e constrói as tags concatenando esses caracteres de acordo com os sinais de abertura e fechamento de tags ("<" e ">"). Sempre que encontra um "<", a flag booleana "inside_tag" recebe true. Quando se está dentro de uma tag, todo caracter diferente de "<" e ">" é concatenado à string "tag". Ao final da tag, quando se encontra um caracter ">", checa-se se a tag encontrada é de abertura ou de fechamento. Se for de abertura, ela é adicionada ao topo da pilha. Caso contrário, deve-se remover o topo da fila e compará-lo à tag. Se a tag atual e o topo da fila não forem correspondentes, ou se a fila estiver vazia, o XML é inválido e retorna-se falso.

3.2 structures Namespace Reference

Namespace que engloba as estruturas utilizadas.

Classes

- class LinkedQueue

  Classe fila encadeada (dinâmica).
- class LinkedStack

  Classe pilha encadeada (dinâmica).
- struct pos_s

  Struct que representa a posição (x, y) de um píxel na imagem.

Typedefs

- typedef struct structures::pos_s pos_t

  Struct que representa a posição (x, y) de um píxel na imagem.

3.2.1 Detailed Description

Namespace que engloba as estruturas utilizadas.

O namespace "structures" contém as classes LinkedQueue e LinkedStack, bem como a estrutura (struct) pos_s, e seu typedef pos_t.

Chapter 4

Class Documentation

4.1 structures::LinkedQueue< T > Class Template Reference

Classe fila encadeada (dinâmica).

`#include <linked_queue.h>`

Public Member Functions

- LinkedQueue ()
  Construtor da fila.
- ~LinkedQueue ()
  Destrutor da fila.
- void clear ()
  Esvazia a fila, removendo todos os elementos.
- void enqueue (const T &data)
  Adiciona um elemento ao final da fila.
- T dequeue ()
  Remove e retorna o primeiro elemento da fila.
- T & front () const
  Retorna o primeiro elemento da fila, sem removê-lo.
- T & back () const
  Retorna o último elemento da fila, sem removê-lo.
- bool empty () const
  Retorna true se a fila estiver vazia, else caso contrário.
- std::size_t size () const
  Retorna o tamanho da fila.

4.1.1 Detailed Description

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

Classe fila encadeada (dinâmica).

The documentation for this class was generated from the following files:

- linked_queue.h
- linked_queue.cpp

4.2 structures::LinkedStack< T > Class Template Reference

Classe pilha encadeada (dinâmica).

`#include <linked_stack.h>`

Public Member Functions

- LinkedStack ()
  Construtor da pilha.
- ~LinkedStack ()
  Destrutor da pilha.
- void clear ()
  Esvazia a pilha, removendo todos os elementos.
- void push (const T &data)
  Adiciona um elemento ao topo da pilha.
- T pop ()
  Remove e retorna (desempilha) o elemento no topo da pilha.
- T & top () const
  Retorna o elemento no topo da pilha, sem removê-lo.
- bool empty () const
  Retorna true se a pilha estiver vazia, else caso contrário.
- std::size_t size () const
  Retorna o tamanho da pilha.

4.2.1 Detailed Description

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

Classe pilha encadeada (dinâmica).

The documentation for this class was generated from the following files:

- linked_stack.h
- linked_stack.cpp

4.3 structures::pos_s Struct Reference

Struct que representa a posição (x, y) de um píxel na imagem.

Public Attributes

- unsigned int **x**
- unsigned int **y**

4.3.1 Detailed Description

Struct que representa a posição (x, y) de um píxel na imagem.

The documentation for this struct was generated from the following file:

- main.cpp

Index

build_matrix
    functions, 5

connected_components
    functions, 6

functions, 5
    build_matrix, 5
    connected_components, 6
    process_xml_file, 6
    verify_xml_file, 7

process_xml_file
    functions, 6

structures, 7
structures::LinkedQueue< T >, 9
structures::LinkedStack< T >, 10
structures::pos_s, 10

verify_xml_file
    functions, 7