Trabalho de Implementação I - processamento de XML com imagens binárias

Disponível até: terça, 9 Out 2018, 12:00
Arquivos requeridos: main.cpp (Baixar)
Número máximo de arquivos: 10
Tipo de trabalho: Trabalho individual

Objetivo

Este trabalho consiste na utilização de estruturas lineares, vistas até o momento no curso, e aplicação de conceitos de pilha e/ou fila para o processamento de arquivos XML contendo imagens binárias. A implementação deverá resolver dois problemas (listados a seguir), e os resultados deverão ser formatados em saída padrão de tela de modo que possam ser automaticamente avaliados no VPL.

Materiais

De modo a exemplificar uma entrada para o seu programa, segue o arquivo XML utilizado no primeiro teste:

- dataset01.xml
  - visualização ampliada das imagens contidas no mesmo:

- dataset02.xml, dataset03.xml, dataset04.xml, dataset05.xml, dataset06.xml
- dicas sobre leitura e escrita com arquivos em C++
  - http://www.cplusplus.com/doc/tutorial/files/
- para a criação e concatenação de palavras/caracteres, sugere-se o uso da classe `string`:
  - http://www.cplusplus.com/reference/string/string/

Primeiro problema: validação de arquivo XML

Para esta parte, pede-se exclusivamente a verificação de aninhamento e fechamento das marcações (*tags*) no arquivo XML (qualquer outra fonte de erro pode ser ignorada). Um identificador (por exemplo: **`img`**) constitui uma marcação entre os caracteres **`<`** e **`>`**, podendo ser de abertura (por exemplo: **`<img>`**) ou de fechamento com uma **`/`** antes do identificador (por exemplo: **`</img>`**). Como apresentando em sala de aula, o algoritmo para resolver este problema é baseado em pilha (**LIFO**):

- Ao encontrar uma marcação de abertura, empilha o identificador
- Ao encontrar uma marcação de fechamento, verifica se o topo da pilha tem o mesmo identificador e desempilha. Aqui duas situações de erro podem ocorrer:
  - Ao consultar o topo, o identificador é diferente (ou seja, uma marcação aberta deveria ter sido fechada antes)
  - Ao consultar o topo, a pilha encontra-se vazia (ou seja, uma marcação é fechada sem que tenha sido aberta antes)
- Ao finalizar a análise (*parser*) do arquivo, é necessário que a pilha esteja vazia. Caso não esteja, mais uma situação de erro ocorre, ou seja, há marcação sem fechamento

Segundo problema: contagem de componentes conexos em imagens binárias representadas em arquivo XML

Cada XML, contém imagens binárias, com altura e largura, definidas respectivamente pelas marcações **`<height>`** e **`<width>`**, e sequência dos pixels (com valores binários, de intensidade 0 para preto ou 1 para branco), em modo texto (embora fosse melhor gravar 1 byte a cada 8 bits, optou-se pelo modo texto por simplicidade), na marcação **`<data>`**. Para cada uma dessas imagens, pretende-se calcular o número de componentes conexos usando vizinhança-4. Para isso, seguem algumas definições importantes:

- A **vizinhança-4** de um pixel na linha ***x*** e coluna ***y***, ou seja, na coordenada **`(x,y)`**, é um conjunto de pixels adjacentes nas coordenadas:
  - **`(x-1, y)`**
  - **`(x+1, y)`**

    - $(x, y-1)$
    - $(x, y+1)$
- Um **caminho** entre um um pixel p_1 e outro p_n é em um sequência de pixels distintos $<p_1, p_2, ..., p_n>$, de modo que p_i é vizinho-4 de p_{i+1}., sendo $i=1,2,...,n-1$
- Um pixel ***p*** é **conexo** a um pixel ***q*** se existir um caminho de ***p*** a ***q*** (no contexto deste trabalho, só há interesse em pixels com intensidade 1, ou seja, brancos)
- Um **componente conexo** é um conjunto maximal (não há outro maior que o contenha) ***C*** de pixels, no qual quaisquer dois pixels selecionados deste conjunto ***C*** são conexos

Para a determinação da quantidade de componentes conexos, antes é necessário atribuir um **rótulo** inteiro e crescente (1, 2, ...) para cada pixel de cada componente conexo. Conforme apresentado em aula, segue o algoritmo de **rotulação** (*labeling*) usando uma fila (**FIFO**):

- Inicializar **rótulo** com **1**
- Criar uma matriz **R** de zeros com o mesmo tamanho da matriz de entrada **E** lida
- Varrer a matriz de entrada **E**
    - Assim que encontrar o primeiro pixel de intensidade **1** ainda não visitado (igual a **0** na mesma coordenada em **R**)
        - inserir **(x,y)** na fila
            - na coordenada **(x,y)** da imagem **R**, atribuir o **rótulo** atual
    - Enquanto a fila não estiver vazia
        - **(x,y)** ← remover da fila
        - inserir na fila as coordenadas dos quatro vizinhos que estejam dentro do domínio da imagem (não pode ter coordenada negativa ou superar o número de linhas ou de colunas), com intensidade **1** (em **E**) e ainda não tenha sido visitado (igual a **0**em **R**)
            - na coordenada de cada vizinho selecionado, na imagem **R**, atribuir o **rótulo** atual
    - incrementar o **rótulo**

O conteúdo final da matriz **R** corresponde ao resultado da rotulação. A quantidade de componentes conexos, que é a resposta do segundo problema, é igual ao último e maior **rótulo** atribuído.

Entrega

- **Individual** ou **em dupla**
- Composição da nota:
    - Nota automática do VPL [(*)]: **70%**
        - [(*)] Caso algum caso de teste não tenha sido bem sucedido, o aluno **opcionalmente** poderá defender sua solução no dia **segunda semana de outubro** reservado à **apresentação do Trabalho de Implementação I**
    - Relatório em PDF (utilize este link para a submissão) com todas as explicações pertinentes e documentação do código [(**)]: **30%**
        - [(**)] Sugere-se a escrita no próprio código usando a notação Doxygen com a geração automática de LaTeX/PDF