Lista #6

Módulo 2 - Hash, Heap e Partição Dinâmica
Data de entrega: 28 de maio de 2017

Acompanham esta lista os arquivos **mod2_lista6.h** e **mod2_lista6.cpp**, contendo as declarações iniciais. Modifique estes arquivos para implementar as questões pedidas e envie-os de volta zipados com nome no padrão <numero_matricula>.zip por email para **profs-eda@tecgraf.puc-rio.br**, com o assunto **[EDA] Lista 6**. Atenção: Crie um arquivo contendo a função main do seu programa para testar suas implementações, mas envie SOMENTE os arquivos e as classes solicitadas.

Contar o número de ocorrências de cada palavra em um texto é uma tarefa que pode ser facilitada com o uso de tabelas Hash. Implemente a classe Hash, cuja especificação se encontra no fim deste documento.

A variável _table é um array dinamicamente alocado, que armazena pares de strings e inteiros, que correspondem às palavras e seus respectivos números de ocorrência. A posição onde deve ser inserida uma palavra é calculada pela função hash, que utiliza a própria palavra como entrada. Assim os elementos podem ser inseridos e consultados em O(1) (caso a função hash também o seja).

Quando ocorrerem colisões (diferentes palavras resultando na mesma saída na função hash), as palavras devem ser armazenadas nas posições conseguintes. Por exemplo, se a palavra "abcd", inserida primeiro, resulta na chave 5 e foi inserida nessa posição, a palavra "efgh", caso resulte também na chave 5, deve ser inserida na posição 6, uma vez que a posição 5 já estava ocupada. E assim sucessivamente.

O método writeCSV deve escrever em arquivo as palavras encontradas no texto, justamente com a sua quantidade de ocorrências. O formato deve ser o CSV (Comma-Separated Values), com cada par palavra-ocorrência em uma linha, e separados por vírgula. Como por exemplo:

casa,2
bola,1
cachorro,1

Observe que entradas vazias (sem palavras) não devem ser impressas.

Caso ache necessário, crie outras funções privadas para auxiliar na inserção ou consulta da tabela.

A seguinte função pode ser utilizada para ler um arquivo de entrada, contabilizar as palavras utilizando a classe Hash implementada, e imprimir a saída em um arquivo. Acompanham dois arquivos com textos para teste, um deles com a sua saída esperada.

```
void countWords( const std::string& inputPath,
                 const std::string& outputPath )
{
    //Abre o arquivo para leitura
    std::ifstream input;
    input.open( inputPath );

    if( input.is_open() ) //Se conseguiu abrir o arquivo
    {
        //Inicializa a Hash
        //Verifique se 512 eh suficiente para o arquivo
        //passado
        Hash wordCounter(512);

        std::string word;
        while( input >> word ) //Le palavra por palavra
        {
            //Insere a palavra na tabela hash
            wordCounter.insert( word );
        }

        //Fecha o arquivo
        input.close();

        //Escreve o resultado em arquivo
        wordCounter.writeCSV( outputPath );
    }
}
```

```
#ifndef HASHTABLE_H
#define HASHTABLE_H

#include <string>

class Hash
{

public:

    /**
     * Construtor. Aloca a tabela com
     * tamanho tableSize
     */
    Hash( int tableSize );

    /**
     * Destrutor. Desaloca a tabela.
     */
    ~Hash();

    /**
      * Insere uma palavra na tabela. Caso ela
      * ja tenha sido inserida, apenas incrementa
      * o numero de ocorrencias.
      */
    void insert(const std::string& word);

    /**
     * Obtem o numero de ocorrencias de uma
     * determinada palavra
     */
    int getCount(const std::string& word);

    /**
     * Escreve todas as palavras e o seu respectivo
     * numero de ocorrencias, no formato CSV (Comma-
     * separated values):
     * palavra,numero_ocorrencias
     *
     * Por exemplo:
     * casa,2
     * bola,1
     * cachorro,1
     *
     * Retorna verdadeiro caso tenha salvo o arquivo
```

```
     * com sucesso.
     */
    bool writeCSV(const std::string& filePath);

    /**
     * Funcao hash. Dada uma palavra, retorna em qual
     * posicao da tabela ela deve ser armazenada
     */
    int hash(const std::string& word);

private:

    // Tabela armazenada como um C-array de pairs.
    // Cada pair eh formado por uma string (palavra)
    // e um inteiro (numero de ocorrencias)
    std::pair< std::string, int >* _table;

};

#endif // HASHTABLE_H
```