fantasy19 / Derp

Code Issues Pull requests Actions Projects Security Insights

e681c03cbb

Derp / 280 / assignment04-BinaryTree / assignment04-BinaryTree / BSTree.h

fantasy19 No commit message

1 contributor

Raw Blame

108 lines (92 sloc) | 3.19 KB

```
/******************************************************************************/
/*!
\file   BSTree.h
\author Ang Cheng Yong
\par    email: a.chengyong\@digipen.edu
\par    DigiPen login: a.chengyong
\par    Course: CS280
\par    Programming Assignment #2
\date   8/11/2016
\brief
This file contains the driver functions needed for BST.
*/
/******************************************************************************/
//---------------------------------------------------------------------------
#ifndef BSTREE_H
#define BSTREE_H
//---------------------------------------------------------------------------
#ifdef _MSC_VER
#pragma warning( disable : 4290 ) // suppress warning: C++ Exception Specification ignored
#endif

#include <string>    // std::string
#include <stdexcept> // std::exception
#include "ObjectAllocator.h"

class BSTException : public std::exception

{
  public:
    BSTException(int ErrCode, const std::string& Message) :
      error_code_(ErrCode), message_(Message) {
    };

    virtual int code(void) const {
      return error_code_;
    }
    virtual const char *what(void) const throw() {
      return message_.c_str();
    }
    virtual ~BSTException() throw() {}
    enum BST_EXCEPTION{E_DUPLICATE, E_NO_MEMORY};

  private:
    int error_code_;
    std::string message_;
};

template <typename T>
class BSTree
{
  public:
    struct BinTreeNode
    {
      BinTreeNode *left;
      BinTreeNode *right;
      T data;
      int balance_factor; // optional
      unsigned count;    // number of nodes in subtree

      BinTreeNode(void) : left(0), right(0), data(0), balance_factor(0), count(0) {};
      BinTreeNode(const T& value) : left(0), right(0), data(value), balance_factor(0), count(0
    };
    typedef BinTreeNode* BinTree;

      BSTree(ObjectAllocator *OA = 0, bool ShareOA = false);
    BSTree(const BSTree& rhs);
    virtual ~BSTree();
    BSTree& operator=(const BSTree& rhs);
    const BinTreeNode* operator[](int index) const;
    virtual void insert(const T& value) throw(BSTException);
    virtual void remove(const T& value);
    void clear(void);
    bool find(const T& value, unsigned &compares) const;
    bool empty(void) const;
    unsigned int size(void) const;
    int height(void) const;
    BinTree root(void) const;

  protected:
    BinTree& get_root(void);
    BinTree make_node(const T& value);
    void free_node(BinTree node);
    int tree_height(BinTree tree) const;
    void find_predecessor(BinTree tree, BinTree &predecessor) const;


  private:
        void free_tree(BinTree & root);
        void copy_tree(BinTree &lhs, BinTree rhs)throw(BSTException);
        void delete_node(BinTree & Tree, const T& value);
        const BinTreeNode* sub_node(BinTree tree, int compares) const;
        void insert_node(BinTree & tree, const T& value) throw(BSTException);
        bool find_node(BinTree tree, const T & value, unsigned& compares) const;

        //unsigned int node_count(BinTree tree) const;
        //unsigned int count;


        ObjectAllocator * oa;
        bool share;
        BinTree root_;
    // private stuff
};

#include "BSTree.cpp"

#endif
//---------------------------------------------------------------------------

```