2022-2 객체지향형 프로그래밍과 자료구조 Exam3C

2022. 11. 19.
담당교수 김영탁

3C. class T_BSTN, class T_BST, BST 재균형화 및 응용 프로그램 (25점)

3C.1 class T_BSTN<E>

- class T_BSTN과 관련 멤버 함수를 T_BSTN.h 파일에 구현할 것

```
/* T_BSTN.h */
template<typename E>
class T_BSTN     // binary search tree node
{
public:
      T_BSTN() : entry(), pPr(NULL), pLc(NULL), pRc(NULL) { } // default constructor
      T_BSTN(E e) : entry(e), pPr(NULL), pLc(NULL), pRc(NULL) { } // constructor
      E& getEntry() { return entry; }
      void setEntry(E e) { entry = e; }
      T_BSTN<E>* getpPr() { return pPr; }
      T_BSTN<E>* getpLc() { return pLc; }
      T_BSTN<E>* getpRc() { return pRc; }
      T_BSTN<E>** getppLc() { return &pLc; }
      T_BSTN<E>** getppRc() { return &pRc; }
      void setpPr(T_BSTN<E>* pTN) { pPr = pTN; }
      void setpLc(T_BSTN<E>* pTN) { pLc = pTN; }
      void setpRc(T_BSTN<E>* pTN) { pRc = pTN; }
      E& operator*() { return entry; }
private:
      E entry;          // element value
      T_BSTN<E>*     pPr;        // parent
      T_BSTN<E>*     pLc;        // left child
      T_BSTN<E>*     pRc;        // right child
};
```

3C.2 class T_BST<E>

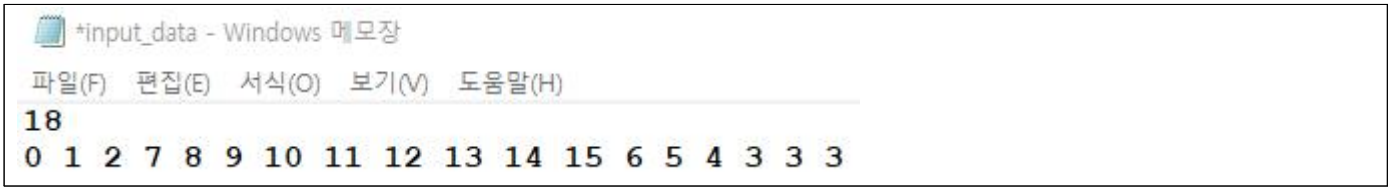
- class T_BST와 관련 멤버 함수를 T_BST.h 파일에 구현할 것

```
/* T_BST.h */

template<typename E>
class T_BST
{
public:
      T_BST(string nm) : _root(NULL), num_entry(0), name(nm) {} // constructor
      string getName() { return name;   }
      int size() const { return num_entry; }
      bool empty() const { return num_entry == 0; }
      T_BSTN<E>* getRoot() { return _root;   }
      T_BSTN<E>** getRootAddr() { return &_root; }
      T_BSTN<E>* eraseBSTN(T_BSTN<E>** pp);
      void insertInOrder(const E entry);
      void insertAndRebalance(E entry);
      void print_with_Depth();
      void print_inOrder();
protected:
      T_BSTN<E>* _insertInOrder(T_BSTN<E>** p, T_BSTN<E>* parenPos, const E e);
      T_BSTN<E>* _insertAndRebalance(T_BSTN<E>** ppTN, T_BSTN<E>* pPr, E e);
      T_BSTN<E>* _rotate_LL(T_BSTN<E> *pCurSubRoot);
      T_BSTN<E>* _rotate_RR(T_BSTN<E> *pCurSubRoot);
      T_BSTN<E>* _rotate_RL(T_BSTN<E> *pCurSubRoot);
      T_BSTN<E>* _rotate_LR(T_BSTN<E> *pCurSubRoot);
      int _getHeight(T_BSTN<E>* pTN);
      int _getHeightDiff(T_BSTN<E>* pTN);
      T_BSTN<E>* _reBalance(T_BSTN<E>** ppTN);
      void _print_with_Depth(T_BSTN<E>* pTN, int depth);
      void _print_inOrder(T_BSTN<E>* pTN);
private:
      T_BSTN<E>* _root; // pointer to the root
      int num_entry; // number of tree nodes
      string name;
};   // end of class T_BST
```

3C.3 입력데이터 파일 준비 (input_data.txt)

- 입력 데이터를 input_data.txt 파일에 준비
- 첫 줄에는 데이터 개수 명시
- 두번째 줄 부터는 데이터를 공란 (space)으로 구분하며 나열



3C.4 class T_BST<E> 기능 시험 프로그램

```
. . .   // include necessary header files and definitions

#include "T_BST.h"
using namespace std;
int* fGetData(string fin_name, int* pNum_data)

{

    // . . . . .  직접 구현할 것

}
void main()
{
        int *data_array;
        int num_data;

        data_array = fGetData(“input_data.txt”, &num_data);
        T_BSTN<int> *pRoot, **ppBST_int_root;
        T_BST<int> BST_int_noBalancing("BST_int_noBalancing");
        cout << "Testing Binary Search Tree without Rebalancing" << endl;
        for (int i = 0; i < num_data; i++)
        {
                BST_int_noBalancing.insertInOrder(data_array[i]);
        }
        BST_int_noBalancing.print_with_Depth();

        cout << "Elements in " << BST_int_noBalancing.getName() << " (in order) : ";
        BST_int_noBalancing.print_inOrder();
        cout << endl;

        T_BST<int> BST_int_Balancing("BST_int_withBalancing");
        cout << "\nTesting Binary Search Tree with Rebalancing" << endl;
        for (int i = 0; i < num_data; i++)
        {
                BST_int_Balancing.insertAndRebalance(data_array[i]);
        }
        BST_int_Balancing.print_with_Depth();

        cout << "Elements in " << BST_int_Balancing.getName() << " (in order) : ";
        BST_int_Balancing.print_inOrder();
        cout << endl;
}
```

3C.4 기능 시험 결과

```
Number of data (from file) = 18
Input data : 0 1 2 7 8 9 10 11 12 13 14 15 6 5 4 3 3 3
Testing Binary Search Tree without Rebalancing
BST_int_noBalancing : current size (18)
                                15
                             14
                          13
                       12
                    11
                 10
              9
           8
        7
           6
              5
                 4
                       3
                    3
                 3
     2
  1
0
Elements in BST_int_noBalancing (in order) : 0 1 2 3 3 3 4 5 6 7 8 9 10 11 12 13 14 15

Testing Binary Search Tree with Rebalancing
BST_int_withBalancing : current size (18)
           15
        14
     13
        12
  11
           10
        9
           8
     7
        6
5
        4
           3
     3
        3
  2
     1
        0
Elements in BST_int_withBalancing (in order) : 0 1 2 3 3 3 4 5 6 7 8 9 10 11 12 13 14 15
```

3C.5 결과물 제출

- 바탕화면의 Exam3 폴더에 Exam3C 프로젝트를 생성
- 압축 파일 내에 포함사항 : 작성한 프로젝트, 실행결과 Capture(채점 시 정확한 실행 유무를 확인하기 위함)
- 실행 화면 캡쳐파일은 각 시험 섹션별로 프로젝트 폴더 내에 저장 후 시험 섹션별 폴더별로 압축
- 제출시 .vs 폴더는 삭제 후 문제별 폴더를 압축하여 제출