Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий комплекс  
«Інститут прикладного системного аналізу»

Відділення другої вищої та післядипломної освіти

**ЛАБОРАТОРНА РОБОТА №1**

(варіант 39)

з курсу «Алгоритми та Структури даних»

на тему «Бінарні дерева»

Виконав: студент 3-го курсу

групи ІС-зп71

Бутузов О.В.

Прийняв: викл. Древаль М.М.

Захищено з оцінкою

« » 2018 р.

Київ – 2018

# Умова завдання

Заповнити бінарне дерево згідно свого варіанту. Провести над ним стандартні для бінарного дерева операції: вставки, видалення, повороту.

# Розв’язання задачі

Для зручної роботи з бінарним деревом був створений окремий клас BunaryTree що підтримував роботу з довільними елементами даних (за допомогою Templates), за основу взято бінарне дерево з дво-направленими вузлами, завдяки чому, хоч і використовується більше місця (4 байта на вузол для вказівника), набагато зручніше працювати з типовими операціями для бінарного дерева.

# Лістинг програм

#include <iostream>               /\* io stream            \*/

#include <string>                 /\* string               \*/

#include <fstream>                /\* files stream         \*/

#include <sstream>                /\* strings stream       \*/

#include "binary-tree.cpp"

#define DEBUG false

#include "view.cpp"

// Type Definition.

typedef std::tuple<int, std::string> Mandragora;

// Overloading Compare Operator

bool operator < ( Mandragora a, Mandragora b );

void loadHarryPotterData(BinaryTree<Mandragora>\* Tree);

int main(){

    BinaryTree<Mandragora> Tree;

    loadHarryPotterData(&Tree);

    std::cout << "Initial Tree\n";

    Tree.tree();

    std::cout << "Deleting Non-Terminal node (23) \n";

    Tree.remove(Tree.find(23));

    Tree.tree();

    std::cout << "Insert New node (23) witha different value. \n";

    Tree.insert(std::make\_tuple(23, std::string("Rufus Scrimgeour") ));

    Tree.tree();

    std::cout << "Left Rotation of node 20 \n";

    Tree.rotate\_left(Tree.find(20));

    Tree.tree();

    std::cout << "Right Rotation of node 25 \n";

    Tree.rotate\_right(Tree.find(25));

    Tree.tree();

    std::cout << "Right Left (Double Right) Rotation of node 28 \n";

    Tree.rotate\_right\_left(Tree.find(28));

    Tree.tree();

    std::cout << "Left Right (Double Left) 8 \n";

    Tree.rotate\_left\_right(Tree.find(8));

    Tree.tree();

    std::cout << "Balancing tree manually \n";

    Tree.rotate\_right(Tree.find(8));

    Tree.rotate\_right(Tree.find(20));

    Tree.rotate\_left\_right(Tree.find(20));

    Tree.tree();

    return 0;

}

/\*\*

 \* Loads Data From File.

 \*/

void loadHarryPotterData(BinaryTree<Mandragora>\* Tree){

    std::ifstream source( "binarytree.txt" );

    if ( !source.good() ){

        throw;

    }

    std::string line;

    while( std::getline( source, line ) ){

        int space\_position( line.find(" ") );

        if ( space\_position == -1 ) {

            continue;

        }

        // Inserting node to Tree

        Tree->insert(

            std::make\_tuple(

                // Node Value

                std::stoi( line.substr( 0, space\_position ) ),

                // Character Name

                line.substr(space\_position+1)

            )

        );

    }

    source.close();

}

/\*\*

 \*  Operator Overload Implementation

 \*  @used in Tree::insert

 \*/

bool operator < ( Mandragora a, Mandragora b ){

    if ( std::get<0>(a) != std::get<0>(b) ) {

        return std::get<0>(a) < std::get<0>(b);

    }

    return false;

}

/\*\*

 \* T value to Int convertor.

 \*/

template <>

int BinaryTree<Mandragora>::value(BinaryTree<Mandragora>::Node\* node) {

    if (node) {

        return std::get<0>(node->data);

    } else {

        return 0;

    }

}

# Результати роботи программи

Результат роботи программи:

**Initial Tree**

.----------015----------------------.

.-008----. .-------------028-.

.-004 .-010-. .-020-------. 029

002 009 011 .-017 .----025-.

016 023-. 026

024

**Deleting Non-Terminal node (23)**

.----------015-------------------.

.-008----. .----------028-.

.-004 .-010-. .-020----. 029

002 009 011 .-017 .-025-.

016 024 026

**Insert New node (23) witha different value.**

.----------015----------------------.

.-008----. .-------------028-.

.-004 .-010-. .-020-------. 029

002 009 011 .-017 .-025-.

016 .-024 026

023

**Left Rotation of node 20**

.----------015----------------------.

.-008----. .----028-.

.-004 .-010-. .-------025-. 029

002 009 011 .-020----. 026

.-017 .-024

016 023

**Right Rotation of node 25**

.----------015----------------------.

.-008----. .-------------028-.

.-004 .-010-. .-020-------. 029

002 009 011 .-017 .-025-.

016 .-024 026

023

**Right Left (Double Right) Rotation of node 28**

.----------015----------------.

.-008----. .-------025----.

.-004 .-010-. .-020----. .-028-.

002 009 011 .-017 .-024 026 029

016 023

**Left Right (Double Left) 8**

.-------015----------------.

.-009-. .-------025----.

.-008 010-. .-020----. .-028-.

.-004 011 .-017 .-024 026 029

002 016 023

**Balancing tree manually**

.-------015----------------.

.----009-. .----------025----.

.-004-. 010-. .-017----. .-028-.

002 008 011 016 .-023-. 026 029

020 024

1. **Висновки**

Наслідком лабораторної роботи, є отримані базові навички з використання структури даних бінарне дерево, знання про базові операції повороту вузлів дерева, вставок та видалень. Отримані знання про переваги та недоліки данної структури данних, їх варіацій.