#include <iostream>

#include <fstream>

#include <Windows.h>

using namespace std;

struct tnode {

int field; // наші дані

struct tnode \*left; // лівий нащадок

struct tnode \*right; // правий нащадок

};

void treeprint(tnode \*tree) { // Прямий обхід дерева, спочатку корінь, потім ліве піддерево і праве піддерево

if (tree!=NULL) { // Поки не зустрінемо пустий вузол

cout << tree->field << " "; // Відображаємо корінь дерева

treeprint(tree->left); // Рекурсивна функція для лівого піддерева

treeprint(tree->right); // Рекурсивна функція для правого дерева піддерева

}

}

tnode \* addnode(int x, tnode \*tree) {

if (tree == NULL) { // Якщо дерева немає, то створюємо корінь

tree = new tnode; // Виділяємо пам'ять під вузол

tree->field = x; // Записуємо дані

tree->left = NULL; // Гілкам присвоюємо NULL

tree->right = NULL; // Гілкам присвоюємо NULL

}else if (x < tree->field) // Умова додавання лівого нащадка

tree->left = addnode(x,tree->left);

else // Умова додавання правого нащадка

tree->right = addnode(x,tree->right);

return(tree);

}

void Reverse(int& number){ // Функція для заміни числа на зворотній йому

bool sign = false; // Покажчик для знаку

string word = to\_string(number); // Конвертуємо число в строку

string result = ""; // Створюємо строку, для результату

for(int i = word.length() - 1; i >= 0; --i){ // Проходимо по стрічці з числом, з кінця

result += word[i]; // І додаємо елементи в нову стрічку

}

if(number < 0) // Якщо наше число від'ємне, то показчику встановлюємо true

sign = true;

number = stoi(result); // Конвертуємо стрічку з результатом в значення int, і присвоюємо значення змінній number

if(sign) // Якщо знак від'ємний

number \*= -1; // То домножуємо number на -1

}

void Replace(tnode\* tree){ // Функція, яка замінює кожний елемент дерева на зворотній

if (tree!=NULL) { // Поки не зустрінемо пустий вузол

Reverse(tree->field); // Використовуємо функцію для заміни числа на зворотній

Replace(tree->left); // Рекурсивна функція для лівого піддерева

Replace(tree->right); // Рекурсивна функція для правого дерева піддерева

}

}

void printleft(tnode\* tree, int &count){ // Функція для лівого піддерева, в якій підщитується кількість парних елементів і виводяться елементи лівого піддерева

if (tree!=NULL) { // Поки не зустрінемо пустий вузол

cout << tree->field << " "; // Виводимо елементи лівого піддерева

printleft(tree->left, count);// Рекурсивна функція для лівого піддерева

if(tree->field % 2 == 0) // Якщо елемент парний, то

++count; // Збільшуємо кількість елементів на 1.

}

}

void printright(tnode\* tree, int &count){ // Функція для правого піддерева, в якій підщитується кількість парних елементів і виводяться елементи правого піддерева

if (tree!=NULL) { // Поки не зустрінемо пустий вузол

cout << tree->field << " "; // Виводимо елементи правого піддерева

printright(tree->right, count); // Рекурсивна функція для правого дерева піддерева

if(tree->field % 2 == 0) // Якщо елемент парний, то

++count; // Збільшуємо кількість елементів на 1.

}

}

void CountGreater(tnode\* tree){ // Функція, яка виводить лівий і правий піддереви окремо і показує, у якого з них більше парних елементів

int count1 = 0, count2 = 0; // Кількість парних елементів лівого і правого піддеревів

cout << "Елементи лівого піддерева: " << endl;

printleft(tree->left, count1); // Виводимо елементи лівого піддерева і присвоюємо змінній count1 кількість парних елементів

cout << endl;

cout << "Кількість парних елементів: " << count1 << endl;

cout << "Елементи правого піддерева: " << endl;

printright(tree->right, count2); // Виводимо елементи правого піддерева і присвоюємо змінній count2 кількість парних елементів

cout << endl;

cout << "Кількість парних елементів: " << count2 << endl;

if(count1 > count2)

cout << "В лівому піддереві кількість парних елементів більша!" << endl;

else if(count2 > count1)

cout << "В правому піддереві кількість парних елементів більша!" << endl;

else

cout << "Кількість парних елементів в обох піддеревах однакова!" << endl;

}

int main(){

SetConsoleCP(1251); // Для українського тексту

SetConsoleOutputCP(1251); // Для українського тексту

tnode\* tree = NULL; // Створюємо дерево

tree = addnode(24, tree); // Записуємо 24, в дерево

tree = addnode(28, tree); // Записуємо 28, в дерево

tree = addnode(44, tree); // Записуємо 44, в дерево

tree = addnode(11, tree); // Записуємо 11, в дерево

tree = addnode(45, tree); // Записуємо 45, в дерево

tree = addnode(-2, tree); // Записуємо -2, в дерево

tree = addnode(-41, tree); // Записуємо -41, в дерево

cout << "Наше дерево виведене прямим обходом: " << endl;

treeprint(tree); // Виводимо дерево на екран

cout << endl;

cout << "Дерево з заміненими елементами на зворотні їм: " << endl;

Replace(tree); // Викликаємо функцію, яка заміняє кожний елемент дерева на зворотній

treeprint(tree); // Виводимо дерево на екран

cout << endl;

CountGreater(tree); // Викликаємо функцію, яка визначає в якому піддереві кількість елементів більша і виводить їх на екран

return 0;

}