#include <cstdlib>

#include <iostream>

using namespace std;

int tabs = 0; // Отступи

int amount = 0; // для парного

int amount\_1 = 0; // для непарного

struct Branch

{

int Data;

Branch\* RightBranch;

Branch\* LeftBranch;

};

void Add(int aData, Branch\*& aBranch)// Додавання до бінарного дерева

{

if (!aBranch)

{

aBranch = new Branch;

aBranch->Data = aData;

aBranch->LeftBranch = 0;

aBranch->RightBranch = 0;

return;

}

else

if (aBranch->Data > aData)

{

Add(aData, aBranch->LeftBranch);

}

else

{

Add(aData, aBranch->RightBranch);

};

}

void print(Branch\* aBranch) // функція виведення дерева

{

if (!aBranch) return;

tabs += 5;

print(aBranch->LeftBranch);

for (int i = 0; i < tabs; i++) cout << " ";

cout << aBranch->Data << endl;

print(aBranch->RightBranch);

tabs-= 5;

return;

}

void pr\_obh(Branch\*& aBranch)

{

if (NULL == aBranch) return;

cout << aBranch->Data << endl; //вузол дерева

pr\_obh(aBranch->LeftBranch); //обхід лівого піддерева

pr\_obh(aBranch->RightBranch); //обхід правого піддерева

}

void gor\_obh(Branch\*& aBranch)

{

if (NULL == aBranch) return;

gor\_obh(aBranch->LeftBranch);

cout << aBranch->Data << endl;

gor\_obh(aBranch->RightBranch);

}

int par(Branch\*& aBranch)

{

if (NULL == aBranch) return 0;

if (aBranch->Data % 2 == 0)

{

amount++;

}

par(aBranch->LeftBranch);

par(aBranch->RightBranch);

return amount;

}

int ne\_par(Branch\*& aBranch)

{

if (NULL == aBranch) return 0;

if (aBranch->Data % 2 == 1 )

{

amount\_1++;

}

ne\_par(aBranch->LeftBranch);

ne\_par(aBranch->RightBranch);

return amount\_1;

}

int summa(Branch\*& aBranch) // сума всіх елементів

{

int sum = 0;

if (aBranch != NULL) {

sum += summa(aBranch->LeftBranch);

sum += aBranch->Data;

sum += summa(aBranch->RightBranch);

}

return sum;

}

void add\_elem(int aData, Branch\*& aBranch) // додавання нового елементу

{

if (!aBranch)

{

aBranch = new Branch;

aBranch->Data = aData;

aBranch->LeftBranch = 0;

aBranch->RightBranch = 0;

return;

}

else

{

if (aData < aBranch->Data) {

add\_elem(aData, aBranch->LeftBranch);

}

else if (aData > aBranch->Data) {

add\_elem(aData, aBranch->RightBranch);

}

}

}

void FreeTree(Branch\* aBranch) // звільнення пам'яті

{

if (!aBranch) return;

FreeTree(aBranch->LeftBranch);

FreeTree(aBranch->RightBranch);

delete aBranch;

return;

}

Branch\* del\_elem(Branch\*& aBranch, int aData) // видалення нового елементу

{

if (aBranch == NULL)

return aBranch;

if (aData == aBranch->Data) {

Branch\* tmp;

if (aBranch->RightBranch == NULL)

tmp = aBranch->LeftBranch;

else {

Branch\* ptr = aBranch->RightBranch;

if (ptr->LeftBranch == NULL) {

ptr->LeftBranch = aBranch->LeftBranch;

tmp = ptr;

}

else {

Branch\* pmin = ptr->LeftBranch;

while (pmin->LeftBranch != NULL) {

ptr = pmin;

pmin = ptr->LeftBranch;

}

ptr->LeftBranch = pmin->RightBranch;

pmin->LeftBranch = aBranch->LeftBranch;

pmin->RightBranch = aBranch->RightBranch;

tmp = pmin;

}

}

delete aBranch;

return tmp;

}

else if (aData < aBranch->Data)

aBranch->LeftBranch = del\_elem(aBranch->LeftBranch, aData);

else

aBranch->RightBranch = del\_elem(aBranch->RightBranch, aData);

return aBranch;

}

int main(){

Branch\* Root = 0;

int num;

int element;

int numbers;

do {

cout << endl;

cout<<" 1. Додати елемент "<<endl;

cout<<" 2. Додати декілька елементів "<<endl;

cout << " 3. Вивести бінарне дерево " << endl;

cout << " 4. Прямий обхiд бiнарного дерева " << endl;

cout << " 5. Симетричний обхiд бiнарного дерева " << endl;

cout << " 6. Видалити елемент " << endl;

cout << " 7. Звільнити пам'ять " << endl;

cout << " 8. Вивести суму всіх елементів " << endl;

cout<<" 0. Вийти\n\n";

cout<<" Номер команди > "; cin>>numbers;

cout << endl;

switch (numbers)

{

case 1:

{

cout << "Введiть елемент: "; cin >> element;

add\_elem(element, Root);

break;}

//-----------------------------------------------

case 2:

{cout << "Кiлькiсть елементiв бiнарного дерева: "; cin >> num;

cout << endl;

for (int i = 0; i < num; i++)

{

Add(rand() % 20, Root); // заповнення цілими числами від 0 до 20

}

break;}

//-----------------------------------------------

case 3:{

print(Root);

cout << endl;

break;}

//-----------------------------------------------

case 4:{ pr\_obh(Root);

cout << endl;

break;

}

//-----------------------------------------------

case 5:{

gor\_obh(Root);

cout << endl;

break;

}

//-----------------------------------------------

case 6:

{

cout << "Введiть елемент для видалення : "; cin >> element;

del\_elem(Root, element);

break;

}

//-----------------------------------------------

case 7:

{

FreeTree(Root);

cout << "Звiльнення динамiчної пам'ятi..." << endl;

break;

}

//-----------------------------------------------

case 8:

{

cout << summa(Root) << endl;

cout << endl;

break;

}

//-----------------------------------------------

case 0: {break;}

default:{ cout<<endl<<" Команда не визначена\n\n";

break;}

}

} while(numbers!= 0);

//system("pause");

return 0;

}