**Домашно 10Б 2020 ДДП. ИБДДП**

**Задача 1.**

Предайте работеща програма (с вход и изход) за образуване на двоично дърво за претърсване. Програмата да докладва след колко проверки установява, че  а) търсената стойност е вече записано в дървото или б) стойността я няма, но се включва като лист.

След последователно въвеждане на стойности, които са: вашия факултетен номер, взет като пет едноцифрени числа и после като две двуцифрени числа, със залепени за него ЕГН - първи четири, взети като едноцифрени и после като двуцифрени (отляво надясно), разпечатайте изход за следното:

* Как изглежда дървото
* Изход от инфиксно обхождане.

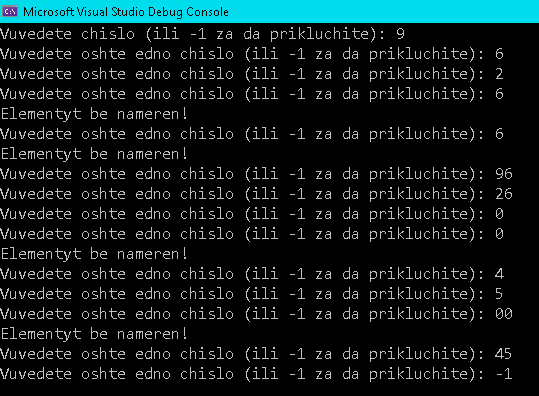
**Задача 2.**

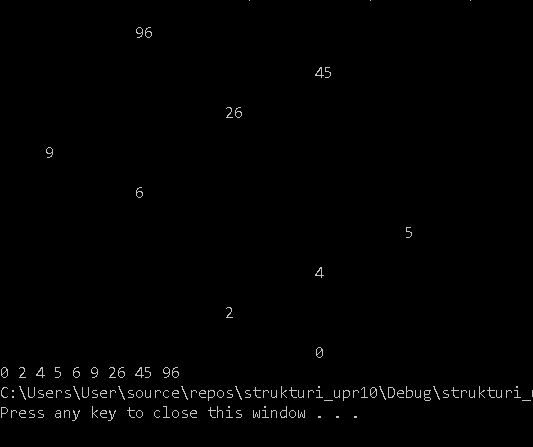
Предайте работеща програма (с вход и изход), която образува Идеално Балансирано Двоично Дълво, но то е Наредено (т.е. то е дърво за претърсване). Програмата получава на входа нареден масив и връща указател към изграденото наредено ИБД.

Въведете на входа **НАРЕДЕНИ** следните стойности, вашия факултетен номер, взет като пет едноцифрени числа и после като две двуцифрени числа, със залепени за него ЕГН - първи четири, взети като едноцрфрени и после като двуцифрени (отляво надясно). (От нас да мине - тази поредица вие получавате от изхода на задача 1. )

Разпечатайте на изход за задача 2 как изглежда ИБД-то.

1.

F96266 egn 0045 - 9,6,2,6,6,96,26,0,0,4,5,00,45



#include <iostream>

using namespace std;

typedef int dataT;

typedef struct node\* Po;

struct node {

dataT data;

Po left;

Po right;

};

void treeBuilder(dataT);

void print(Po, int);

void Infix(Po);

Po root = nullptr;

int main() {

int number;

cout << "Vuvedete chislo (ili 0 za da prikluchite): ";

cin >> number;

while (number != 0)

{

treeBuilder(number);

cout << "Vuvedete oshte edno chislo (ili 0 za da prikluchite): ";

cin >> number;

}

print(root, 5);

Infix(root);

return 0;

}

void treeBuilder(dataT x) {

Po loc = root;

Po insert = loc;

if (root == nullptr)

{

loc = new node;

loc->left = nullptr;

loc->right = nullptr;

loc->data = x;

root = loc;

}

else {

/\*

loc trqbva da spre kogato found == true

ILI

loc == nullptr

\*/

while (loc != nullptr && loc->data != x)

{

insert = loc;

if (loc->data > x)

{

loc = loc->left;

}

else

{

loc = loc->right;

}

}

if (loc != nullptr)

{

cout << "Elementyt be nameren!" << endl;

}

else {

loc = new node;

loc->data = x;

loc->left = nullptr;

loc->right = nullptr;

if (loc->data > insert->data)

{

insert->right = loc;

}

else {

insert->left = loc;

}

}

}

}

void print(Po root, int space) {

int COUNT = 10;

if (root == nullptr)

return;

space += COUNT;

print(root->right, space);

std::cout << std::endl;

for (int i = COUNT; i < space; i++)

std::cout << " ";

std::cout << root->data << "\n";

print(root->left, space);

}

void Infix(Po help)

{

if (help)

{

Infix(help->left);

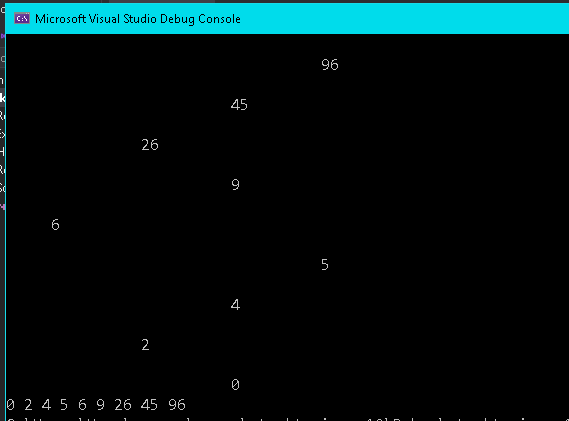
cout << help->data << " ";

Infix(help->right);

}

}

зад 2



#include <iostream>

using namespace std;

typedef int dataT;

typedef struct node\* Po;

struct node {

dataT data;

Po left;

Po right;

};

Po root = nullptr;

void print(Po root, int space) {

int COUNT = 10;

if (root == nullptr)

return;

space += COUNT;

print(root->right, space);

std::cout << std::endl;

for (int i = COUNT; i < space; i++)

std::cout << " ";

std::cout << root->data << "\n";

print(root->left, space);

}

void Infix(Po help)

{

if (help)

{

Infix(help->left);

cout << help->data << " ";

Infix(help->right);

}

}

Po TreeFromSortedArr(int\* arr, int beg, int end) {

if (beg <= end) {

int mid = (beg + end) / 2;

Po help = new node;

help->data = arr[mid];

help->left = nullptr;

help->right = nullptr;

help->left = TreeFromSortedArr(arr, beg,

mid - 1);

help->right = TreeFromSortedArr(arr, mid + 1, end);

return help;

}else

{

return 0;

}

}

int main() {

int arr[9] = {0,2,4,5,6,9,26,45,96};

//int end = sizeof(arr)/ sizeof(arr[0]);

Po root = TreeFromSortedArr(arr, 0, 8);

print(root, 5);

Infix(root);

return 0;

}