 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Лабораторна робота №12**

з дисципліни «Алгоритмізація та програмування

на тему: «Дерева»

Варіант №

**Виконав:**

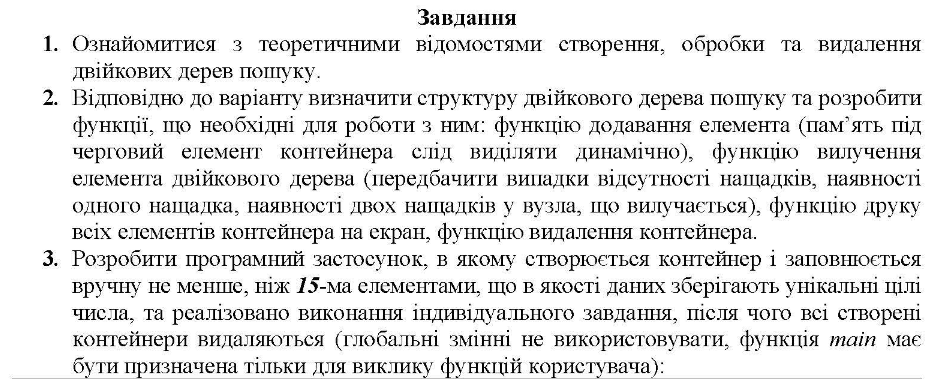
**Перевірив:**

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2022

Код програми:

#include <iostream>

#include<cstdlib>

using namespace std;

struct Node

{

int data;

Node\* right;

Node\* left;

};

Node\* Ins(Node\* root, int data)

{

if (root == nullptr)

{

root = new Node;

root->data = data;

root->right = nullptr;

root->left = nullptr;

return root;

}

if (data == root->data)

return nullptr;

if (data < root->data)

root->left = Ins(root->left, data);

if (data > root->data)

root->right = Ins(root->right, data);

return root;

}

void Display(int level, Node\* root)

{

int i;

if (root != nullptr)

{

Display(level + 1, root->right);

cout << endl;

for (i = 0; i < level; i++)

cout << " ";

cout << root->data;

Display(level + 1, root->left);

}

}

Node\* Del(Node\* root, int data)

{

if (root == nullptr)

return root;

if (data < root->data)

root->left = Del(root->left, data);

else

if (data > root->data)

root->right = Del(root->right, data);

else

{

if (root->left == nullptr && root->right == nullptr)

{

delete root;

root = nullptr;

return root;

}

if (root->left == nullptr)

{

Node\* tmp = root;

root = root->right;

delete tmp;

return root;

}

if (root->right == nullptr)

{

Node\* tmp = root;

root = root->left;

delete tmp;

return root;

}

Node\* tmp = root->right;

while (tmp->left != nullptr)

tmp = tmp->left;

root->data = tmp->data;

root->right = Del(root->right, tmp->data);

}

return root;

}

void DelAll(Node\* root)

{

if (root->right != nullptr)

DelAll(root->right);

if (root->left != nullptr)

DelAll(root->left);

delete root;

root = nullptr;

}

int main()

{

Node\* root = nullptr;

int i, a;

cout << "How much elements do you want to be in binary tree?: ";

cin >> i;

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

{

int data;

cout << "Enter data: ";

cin >> data;

root = Ins(root, data);

}

cout << endl << endl << "Given tree: ";

Display(0, root);

cout << endl << "Enter what you want to delete: ";

cin >> a;

Del(root, a);

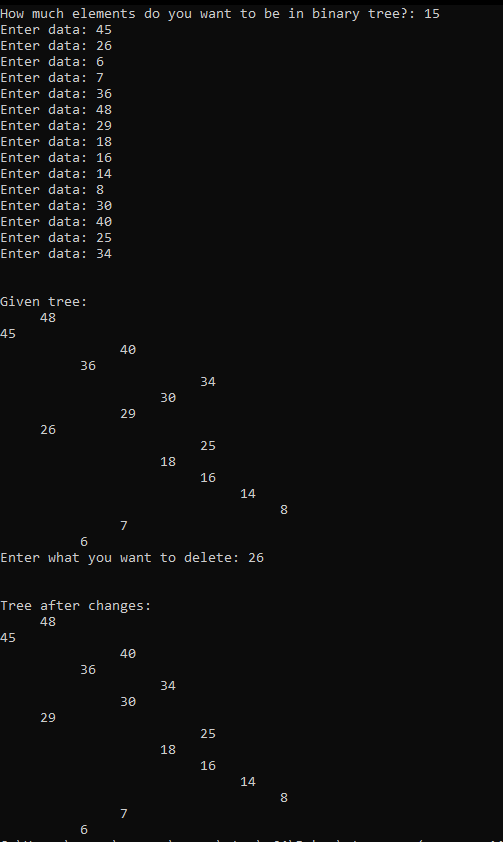
cout << endl << endl << "Tree after changes: ";

Display(0, root);

DelAll(root);

return 0;

}

Результати виконання: 

**Контрольні питання**

**1. Що таке дерево? В чому його відмінність від списку?**

Дерево – це сукупність елементів, які називаються вузлами(при цьому один з них визначається як корінь), і відносин(батько-нащадок), що утворюють ієрархічну структуру вузлів, які мають відношення «один до багатьох».

За допомогою дерева можна описати ієрархічні структури, що подібні до каталогів файлів, з списками це складно зробити.

**2. Які бувають вузли дерева? Шо таке глибина дерева?**

Вузол з самого верху називається коренем, для нижнього вузла він називається батьком.

Вузли що не мають нащадків називаються термінальними вузлами або листям дерева.

Нетермінальні вузли називаються внутрішніми вузлами дерева.

Глибина вузла є числом ребер від вузла до кореневого дерева. Кореневий вузол матиме глибину 0.

**3. Який тип інформаційних структур зручно представляти за допомогою дерев? Наведіть приклади.**

Ієрархічні структури, що подібні до каталогів файлів.

**4. Що таке обхід дерева? Який спосіб виклику функцій для нього використовується?**

Обхід дерева – це спосіб дослідження вузлів дерева, при якому кожен вузол відвідується тільки один раз. При цьому в результаті обходу дерева виходить лінійна розстановка вузлів. Використовується рекурсивний спосіб виклику.

**5. Які існують види обходу дерев? Приведіть довільне невироджене дерево з не меньш, ніж 8- ма елементами та не меньш, ніж 3-ма рівнями та опишіть варіанти його обходу.**

Залежно від траєкторії виділяють два типи обходу:

* Горизонтальний
* Вертикальний

Під горизонтальним мається на увазі обхід дерева за рівнями. ПІд час такгго обходу спочатку відвідуються всі вузли поточного рівня, після чого здійснюється перехід на нижній рівень.

Під час вертикального обходу порядок роботи поточного вузла і вузлів його лівого та правого піддерев можуть бути різними. За цією ознакою виділяють три варіанти вертикального обходу, кожний з яких визначається рекурсивно:

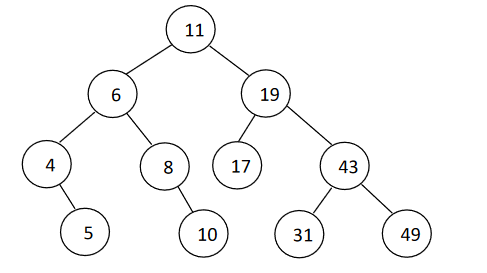
- прямий порядок

- зворотний порядок

- симетричний порядок  
Прямий порядок передбачає виконання наступної послідовності кроків: відвідати корінь – відвідати ліве піддерево – відвідати праве піддерево. Тобто в такому порядку обходу кожна вершина відвідується до того, як будуть відвідані її діти.

Зворотний порядок передбачає виконання наступної послідовності кроків: відвідати ліве піддерево – відвідати праве піддерево – відвідати корінь. Тобто в такому порядку кожна вершина відвідується лише після того, як будуть відвідані її діти.

Симетричний порядок передбачає виконання наступної послідовності кроків: відвідати ліве піддерево – відвідати корінь – відвідати праве піддерево. У такому порядку кожна вершина відвідується між відвіданням лівої та правої дитини. Такий порядок особливо часто застосовується в двійкових деревах пошуку, тому що дає можливість обходу вершин у порядку збільшення їхнії порядкових номерів.



Прямий: 11-6-4-5-8-10-19-17-43-31-49

Зворотній: 5-4-10-8-6-17-31-49-43-19-11

Симетричний: 4-5-6-8-10-11-17-19-31-43-49

**6. Дайте визначення двійковому дереву та двійковому дереву пошуку.**

Двійковим деревом називають дерево, у кожного вузла якого є не більше двох нащадків.

Двійкове дерево пошуку – це таке дерево, в якому значення всіх лівих нащадків менші за значення, що зберігає їх батьківський вузол, а значення всіх правих нащадків – більші.

**7. Що таке повне та досконале дерева та в чому їх відмінність?**

У повному бінарному дереві кожен вузол має рівно 0 або 2 дочірніх.

Досконале бінарне дерево – таке бінарне дерево, у якого (h-1) рівнів є повними, а рівень h (на якому всі вузли– листи) має або тільки ліві вузли, або з двома нащадками.

Повне дерево також є досконалим деревом.

**8. Чи можна побудувати повне двійкове дерево з елементів «5», «8», «1», «6», «2», «10», «3», «9», «4», «7»?**

Ні, тому що будуть вузли з одним нащадком.

**9. Наведіть алгоритм видалення елемента двійкового дерева у випадку, коли цей елемент не має нащадків, має одного нащадка, має двох нащадків.**

У цілому, алгоритм видалення елемента виглядає так:

* Знайти вузол, який потрібно видалити.
* Видалити його.

Випадок 1: У вузла нема нащадків. Треба тільки видалити вузол.

Випадок 2: У вузла, що видаляється, 1 нащадок. Треба видалити вузол, нащадка помістити на місце вузла.

Випадок 3: : У вузла, що видаляється, 2 нащадка. Треба видалити вузол і на його місце поставити самий лівий вузол піддерева правого нащадка. Елемент, яким замінили видалений треба видалити з місця, в якому він знаходився.

**10. Елементом яких деревовидних структур є кожна людина**

Сімейне дерево.