 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №17**

з дисципліни «Алгоритмізація та програмування»

на тему: «Дерева»

Варіант №2

**Виконав:**

студент гр. БС-71

Батуркіна А.М.

**Перевірив:**

ас. каф. БМК

Рисін С. В.

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

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

(підпис викладача)

Київ-2018

**🞏 Комп’ютерний практикум без зауважень**

**🞏 Комп’ютерний практикум має зауваження:**

**🞏 несвоєчасний захист**

**🞏 присутні зауваження до блок-схеми:**

**🞏 блок-схема не відповідає коду**

**🞏 в блок-схемі присутній код**

**🞏 виконані не за стандартом:**

**🞏 блок умови 🞏 визначений процес (функція)**

**🞏 оператор вибору 🞏 перехід**

**🞏 цикл 🞏 розміри блоків**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 присутні зауваження до коду:**

**🞏 задача завдання вирішена хибно**

**🞏 код програми не компілюється**

**🞏 використано глобальні змінні**

**🞏 типи даних визначені хибно**

**🞏 недостатня декомпозиція на функції користувача**

**🞏 функція main містить лише виклик іншої функції**

**🞏 статичні змінні при роботі з масивами**

**🞏 оформлення коду**

**🞏 присутні зайві символи «{» та «}»**

**🞏 інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**🞏 результати виконання програми на рисунках не відповідають коду**

**🞏 невірні відповіді на запитання:**

**🞏 №1 🞏 №2 🞏 №3 🞏 №4 🞏 №5**

**🞏 №6 🞏 №7 🞏 №8 🞏 №9 🞏 №10**

**🞏 незнання теоретичного матеріалу**

**🞏 маються інші зауваження:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

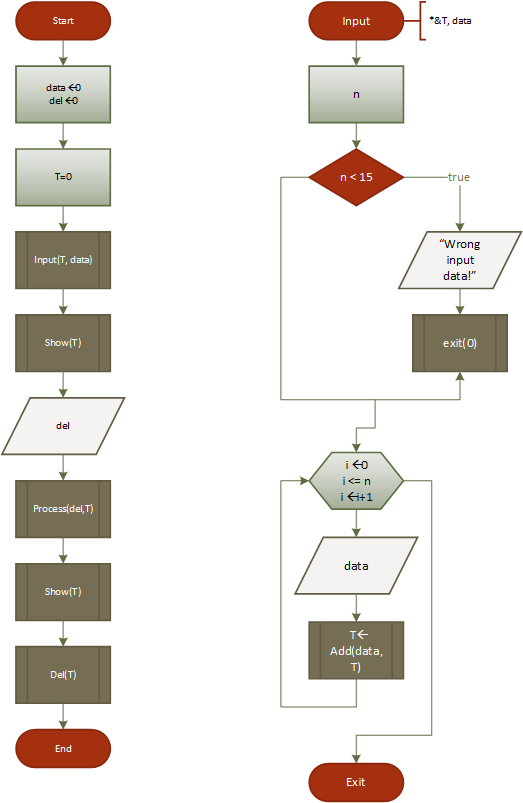
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

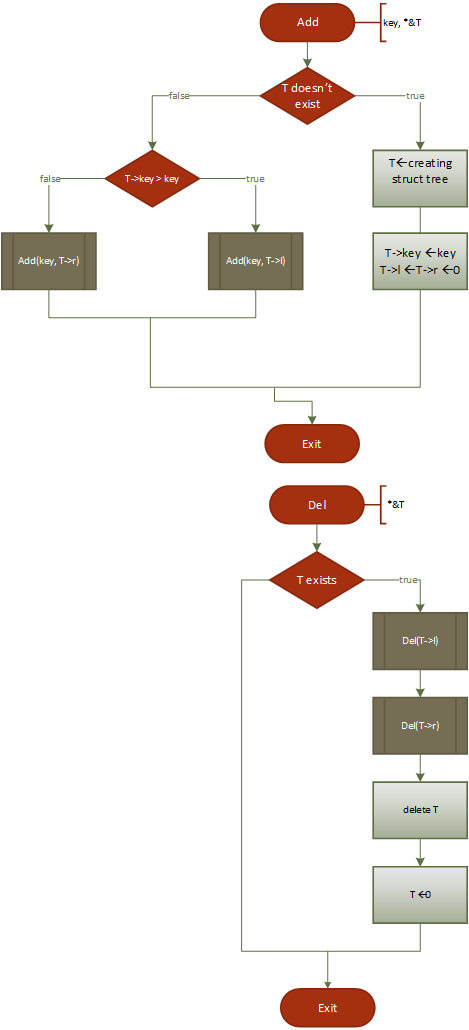
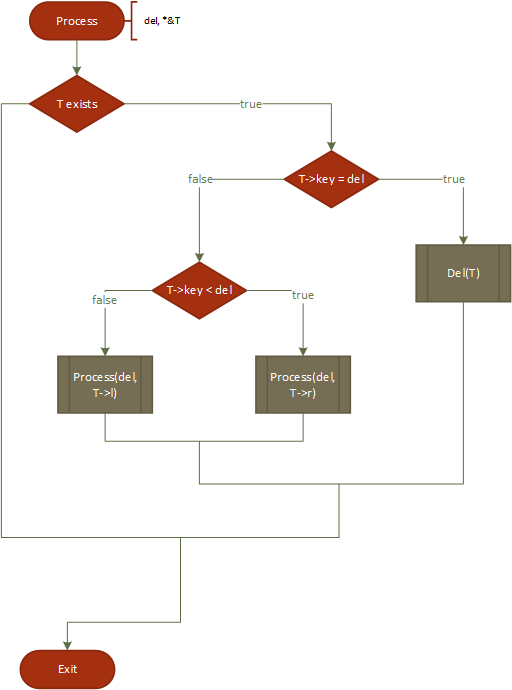
**Завдання:**

1. Ознайомитися з теоретичними відомостями створення, обробки та видалення двійкових дерев пошуку.
2. Відповідно до варіанту визначити структуру двійкового дерева пошуку та розробити функції, що необхідні для роботи з ним: функцію додавання елемента (пам’ять під черговий елемент контейнера слід виділяти динамічно), функцію друку всіх елементів контейнера на екран, функцію видалення контейнера.
3. Побудувати блок-схему алгоритму для вирішення задачі відповідно до свого варіанту.
4. Розробити програмний застосунок, в якому створюється контейнер і заповнюється вручну не менше, ніж 15-ма елементами, що в якості даних зберігаються унікальні цілі числа, та реалізовано виконання індивідуального завдання, після чого всі створені контейнери видаляються (глобальні змінні не використовувати, функція main має бути призначена тільки для виклику функцій користувача):

*Видалити піддерево з вершиною, що має заданий ключ. Ключ задати з клавіатури.*

1. Скласти та захистити звіт по роботі.

****Блок-схеми:**

**

**Лістинг:**

#include <iostream>

#include <conio.h>

using namespace std;

struct tree

{

int key;

tree \*l, \*r;

};

void Add(int key, tree \*&T)

{

if (!T)

{

T = new tree; T->key = key; T->l = T->r = 0;

}

else

if (T->key > key)

Add(key, T->l);

else

Add(key, T->r);

}

void Show(tree \*T)

{

if (T)

{

Show(T->l);

cout << T->key << " ";

Show(T->r);

}

}

void Del(tree \*&T)

{

if (T)

{

Del(T->l);

Del(T->r);

delete T;

T = 0;

}

}

void Process(int del, tree \*&T)

{

if (T != NULL)

{

if (T->key == del)

Del(T);

else if (T->key < del)

Process(del, T->r);

else

Process(del, T->l);

}

}

void Input(tree \*&T, int data)

{

int n;

cout << "Input number of elements(no less than 15!):";

cin >> n;

if (n < 15)

{

cout << "Wrong input number of elements!";

\_getch();

exit(0);

}

for (int i = 1; i <= n; i++)

{

cout << "Input unique " << i <<" element: ";

cin >> data;

Add(data, T);

}

cout << endl;

}

int main()

{

int data = 0;

tree \*T = 0;

int del;

Input(T, data);

Show(T);

cout << endl << endl;

cout << "Input key of element you'd like to delete: ";

cin >> del;

cout << endl;

Process(del, T);

Show(T);

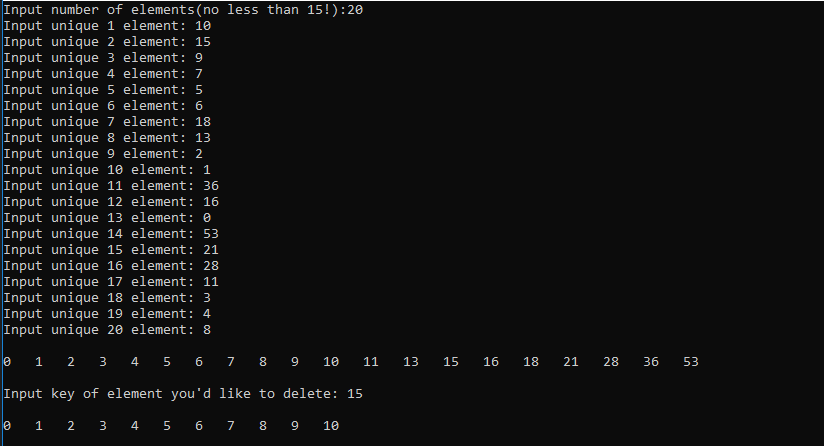
Del(T);

\_getch();

return 0;

}

**Результати:**

****

**Контрольні запитання:**

1. Дайте визначення дереву.

* Дерево – зв’язний неорієнтований граф, що не містить циклів.

1. Що таке двійкове дерево, двійкове дерево пошуку?

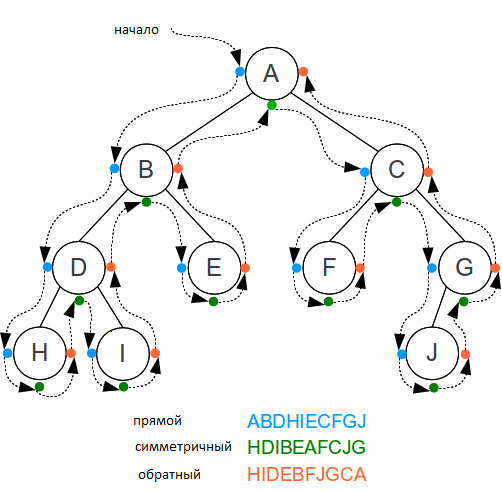
* Двійкове дерево – це контейнер, який є програмною реалізацією графа двійкового дерева та складається з вузлів(вершин), які мають не більше двох дітей(нащадків)(тобто має запис виду data,r,l).
* Двійкове дерево пошуку – тип двійкового дерева, для кожної вершини якого виконуються 2 умови: 1) значення вершини більше за значення будь-якої вершини його лівого піддерева; 2) значення вершини менше за значення будь-якої з вершини його правого піддерева.

1. Який тип інформації зручно представляти за допомогою двійкових дерев?

* Ієрархічні структури, подібні до каталогів файлів.

1. Що таке обхід дерева, коли його застосовують?

* Обхід дерева – це спосіб дослідження вузлів дерева, при якому кожен вузол проходиться точно один раз. Існують три види обходу: прямий, зворотний, симетричний.
* Обхід використовується для перегляду елементів дерева та виконування операцій над деревом. Наприклад, пошук, видалення, додавання елементів.

1. Навести схему довільного двійкового дерева пошуку та вказати порядок прямого, зворотного та симетричного обходу. 
2. Які алгоритми пошуку елементів зручно застосувати під час роботи з деревами?

- алгоритми пошуку є 1) в глибину та 2) в ширину

1. ЧИ можна побудувати повне двійкове дерево з елементів 5, 1, 2 , 8, 6, 10, 3, 9, 4, 7? Відповідь пояснити.

* Не можна, тому що кількість вузлів повинна бути непарною.

1. Що таке досконале дерево, чим воно відрізняється від повного?

* Досконале дерево - це таке повне двійкове дерево, в якому листя (вершини без дітей) лежать на однаковій глибині (відстані від кореня).
* Повне двійкове дерево — таке двійкове дерево, в якому кожна вершина має нуль або двох дітей.

1. Поясніть принцип роботи рекурсивних функцій для відображення елементів двійкового дерева.

* Спочатку на вхід функції подається корінь і рекурсивно визиває лівий нащадок, потім відображається і рекурсивно визиває правий.

1. Поясніть принцип видалення елемента двійкового дерева у випадку, коли цей елемент не має нащадків, має одного нащадка, має двох нащадків.

* Не має нащадків : в такому випадку видалення для форми дерево зводиться до простої заміни покажчика в батьківському вузлі на нульове значення.
* Має одного нащадка : в даному випадку батько видаляємого елемента всиновляє його єдиний дочірній елемент.
* Має двох нащадків : в даному випадку нащадком видаляємого вузла буде найлівіший вузол у правому піддереві.