

Лабораторна робота № 9

Робота з бінарними деревами

Мета роботи: набуття практичних навичок побудови бінарних дерев та їх використання для розв'язування практичних задач.

Хід роботи

Завдання для виконання

1. Виконати Завдання 1 та Завдання 2 та продемонструвати побудоване дерево, здійснивши його обхід у глибину та ширину. Результати роботи виводити на екран після кожної операції та відобразити у звіті.

Завдання 1. Реалізуйте клас бінарного дерева, використовуючи рекурсивний підхід та створіть таке дерево із літер Вашого прізвища, імені та по батькові.

Завдання 2. Реалізуйте алгоритми пошуку в глибину та ширину у бінарному дереві, яке було створене у Завданні 1. Результати виведіть та відобразіть у звіті.

Лістинг 9.1 – Код програми

```
#include <iostream>
#include <string>
using namespace std;
struct n {
    char data;
    n* left;
    n* right;

    n(char c) : data(c), left(nullptr), right(nullptr) {}
};
class tree {
public:
    n* root;
    tree() : root(nullptr) {}

    void r(char c) {
        if (root == nullptr) {
            root = new n(c);
            return;
        }
        q(root, c);
    }
    void q(n* node, char c) {
        if (node->left == nullptr) {
```

```
        node->left = new n(c);
        return;
    }
    else if (node->right == nullptr) {
        node->right = new n(c);
        return;
    }
    else {
        q(node->left, c);
    }
}
void p{
    if (!node) return;
    cout << node->data << " ";
    p(node->left);
    p(node->right);
}

void t(n* node) {
    if (!node) return;

    n* queue[100];
    int front = 0, back = 0;

    queue[back++] = node;

    while (front < back) {
        n* cur = queue[front++];
        cout << cur->data << " ";

        if (cur->left) queue[back++] = cur->left;
        if (cur->right) queue[back++] = cur->right;
    }
}
};
int main() {
    cout << "Введіть прізвище, ім'я та по батькові: ";
    string s;
    getline(cin, s);
    string letters;
    for (char c : s) {
        if (c != ' ') letters.push_back(c);
    }
    tree tree;
    for (char c : letters) {
        tree.r(c);
    }
    cout << "\n06хід у глибину (preorder): ";
    tree.p(tree.root);
    cout << "\n06хід у ширину (breadth-first): ";
    tree.t(tree.root);
    cout << "\n";
    return 0;
}
```

}

```

Введіть прізвище, ім'я та по батьку: Чавурська Єлизавета Віталіївна
Обхід у глибину (preorder): 
Обхід у ширину (breadth-first):

```

Рисунок 9.1 – Вікно результатів програми

Лістинг 9.1 – Код програми 2

```

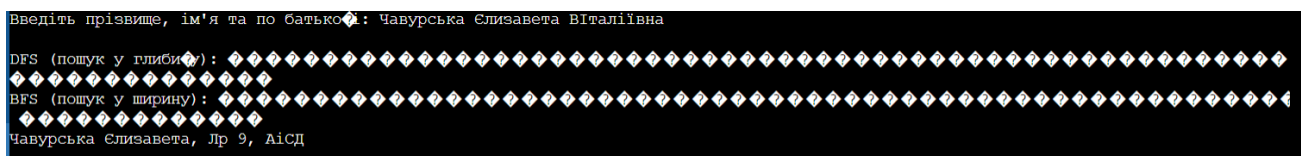
#include <iostream>
#include <string>
using namespace std;
struct Node {
    char data;
    Node* left;
    Node* right;
    Node(char c) : data(c), left(nullptr), right(nullptr) {}
};
class c {
public:
    Node* root;
    c() : root(nullptr) {}
    void a(char ch) {
        if (root == nullptr) {
            root = new Node(ch);
            return;
        }
        b(root, ch);
    }
    void b(Node* node, char ch) {
        if (node->left == nullptr) {
            node->left = new Node(ch);
            return;
        }
        else if (node->right == nullptr) {
            node->right = new Node(ch);
            return;
        }
        else {
            b(node->left, ch);
        }
    }
    void DFS(Node* node) {
        if (!node) return;
        cout << node->data << " ";
        DFS(node->left);
        DFS(node->right);
    }
    void BFS(Node* node) {

```

```
    if (!node) return;
    Node* queue[100];
    int front = 0, back = 0;
    queue[back++] = node;
    while (front < back) {
        Node* cur = queue[front++];
        cout << cur->data << " ";
        if (cur->left) queue[back++] = cur->left;
        if (cur->right) queue[back++] = cur->right;
    }
};
int main() {
    cout << "Введіть прізвище, ім'я та по батькові: ";
    string s;
    getline(cin, s);

    string letters;
    for (char c : s) {
        if (c != ' ') letters.push_back(c);
    }
    c tree;

    for (char ch : letters) {
        tree.a(ch);
    }
    cout << "\nDFS (пошук у глибину): ";
    tree.DFS(tree.root);
    cout << "\nBFS (пошук у ширину): ";
    tree.BFS(tree.root);
    cout << "\n";
    cout << "Чавурська Єлизавета, Лр 9, AiCD";
    return 0;
}
```



```
Введіть прізвище, ім'я та по батькові: Чавурська Єлизавета Віталіївна
DFS (пошук у глибину): 
BFS (пошук у ширину): 
Чавурська Єлизавета, Лр 9, AiCD
```

Рисунок 9.2 – Вікно результатів програми

2. Виконати Завдання 3 та продемонструвати результати роботи програми здійснюючи обхід в ширину. Результати роботи виводити на екран після кожної операції та відобразити у звіті.

Завдання 3. Доповніть створений клас реалізацією операцій пошуку та вставки (рекурсивно, чи нерекурсивно – за Вашим бажанням).

Перевірте, чи наявні у Вашому дереві літери А, О, К, Ф, Ю, Ї.

Результати перевірки виведіть на екран та відобразіть у звіті. Вставте у Ваше дерево літери Я, Ж, И, П.

Лістинг 9.3 – Код програми

```
#include <iostream>
#include <queue>
#include <string>
using namespace std;

struct Node {
    string data;
    Node* left;
    Node* right;
    Node(string d) : data(d), left(NULL), right(NULL) {}
};

class Tree {
public:
    Node* root;

    Tree() : root(NULL) {}

    void a(Node*& node, const string& value) {
        if (!node) {
            node = new Node(value);
            return;
        }
        if (value < node->data) a(node->left, value);
        else a(node->right, value);
    }

    bool b(Node* node, const string& value) {
        if (!node) return false;
        if (node->data == value) return true;
        if (value < node->data) return b(node->left, value);
        return b(node->right, value);
    }

    void c(Node* node) {
        if (!node) return;
        queue<Node*> q;
        q.push(node);
        while (!q.empty()) {
            Node* cur = q.front();
            q.pop();
        }
    }
};
```

```
        cout << cur->data << " ";
        if (cur->left) q.push(cur->left);
        if (cur->right) q.push(cur->right);
    }
}

};

int main() {
    Tree t;

    string letters[] = { "Т", "В", "Р", "Г", "О" };
    for (auto& s : letters) t.a(t.root, s);

    cout << "Початкове дерево (BFS): ";
    t.c(t.root);
    cout << endl;

    string checkLetters[] = { "А", "О", "К", "Ф", "Ю", "Ь" };
    for (auto& s : checkLetters) {
        cout << "Пошук '" << s << "': ";
        cout << (t.b(t.root, s) ? "є" : "нема") << endl;
    }

    string insertLetters[] = { "Я", "Ж", "И", "П" };
    for (auto& s : insertLetters) {
        t.a(t.root, s);
        cout << "Після вставки '" << s << "' (BFS): ";
        t.c(t.root);
        cout << endl;
    }

    cout << "Чавурська Єлизавета, П-31, AiCD, ЛР 9";
    return 0;
}
```

```
Початкове дерево (BFS): Т В Р Г О
Пошук 'А': нема
Пошук 'О': є
Пошук 'К': нема
Пошук 'Ф': нема
Пошук 'Ю': нема
Пошук 'Ь': нема
Після вставки 'Я' (BFS): Т В Я Р Г О
Після вставки 'Ж' (BFS): Т В Я Р Г О Ж
Після вставки 'И' (BFS): Т В Я Р Г О Ж И
Після вставки 'П' (BFS): Т В Я Р Г О Ж П И
Чавурська Єлизавета, П-31, AiCD, ЛР 9
```

Рисунок 9.3 – Вікно виконання програми

3. Виконати Завдання 4 та перевірити роботу алгоритму аналогічно до Завдання 3. Результати роботи виводити на екран після кожної операції та відобразити у звіті.

Завдання 4. Доповніть створений клас реалізацією операції видалення вузла дерева.

Перевірте роботу, видаливши з дерева всі літери, що співпадають із початковими літерами Вашого прізвища, імені та по батькові. Результати відобразіть у звіті.

Лістинг 9.4 – Код програми

```
#include <iostream>
#include <queue>
#include <string>
using namespace std;

struct Node {
    string data;
    Node* left;
    Node* right;
    Node(string d) : data(d), left(NULL), right(NULL) {}
};

class Tree {
public:
    Node* root;

    Tree() : root(NULL) {}

    void a(Node*& node, const string& value) {
        if (!node) {
            node = new Node(value);
            return;
        }
        if (value < node->data) a(node->left, value);
        else a(node->right, value);
    }

    bool b(Node* node, const string& value) {
        if (!node) return false;
        if (node->data == value) return true;
        if (value < node->data) return b(node->left, value);
        return b(node->right, value);
    }
}
```

```
Node* minNode(Node* node) {
    while (node && node->left) node = node->left;
    return node;
}

Node* f(Node* node, const string& value) {
    if (!node) return NULL;

    if (value < node->data) node->left = f(node->left, value);
    else if (value > node->data) node->right = f(node->right,
value);
    else {
        if (!node->left) {
            Node* r = node->right;
            delete node;
            return r;
        }
        if (!node->right) {
            Node* l = node->left;
            delete node;
            return l;
        }
        Node* m = minNode(node->right);
        node->data = m->data;
        node->right = f(node->right, m->data);
    }
    return node;
}

void c(Node* node) {
    if (!node) {
        cout << "(порожнє дерево)";
        return;
    }
    queue<Node*> q;
    q.push(node);
    while (!q.empty()) {
        Node* cur = q.front();
        q.pop();
        cout << cur->data << " ";
        if (cur->left) q.push(cur->left);
        if (cur->right) q.push(cur->right);
    }
}

};

int main() {
    Tree t;

    string letters[] = { "T", "B", "P", "Г", "O" };
    for (auto& s : letters) t.a(t.root, s);
}
```

```

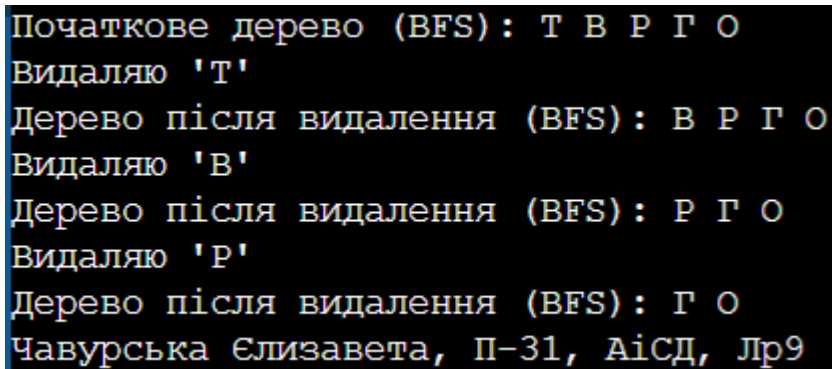
cout << "Початкове дерево (BFS): ";
t.c(t.root);
cout << endl;

string d[] = { "Т", "В", "Р" };

for (auto& s : d) {
    cout << "Видаляю '" << s << "'" << endl;
    t.root = t.f(t.root, s);
    cout << "Дерево після видалення (BFS): ";
    t.c(t.root);
    cout << endl;
}
cout << "Чавурська Єлизавета, П-31, AiCD, Лр9";

return 0;
}

```



```

Початкове дерево (BFS): Т В Р Г О
Видаляю 'Т'
Дерево після видалення (BFS): В Р Г О
Видаляю 'В'
Дерево після видалення (BFS): Р Г О
Видаляю 'Р'
Дерево після видалення (BFS): Г О
Чавурська Єлизавета, П-31, AiCD, Лр9

```

Рисунок 9.5 – Вікно результатів програми

4. Виконати індивідуальне Завдання 5 згідно додатку 1.

Додаток 1

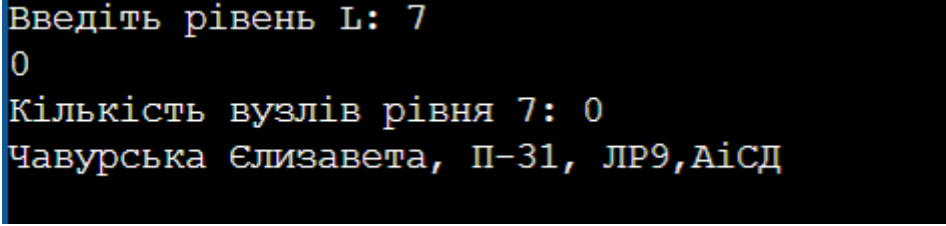
Варіант	Умова завдання
20	З вихідної послідовності дійсних чисел побудувати бінарне дерево пошуку. Використовуючи будь який із способів обходу дерева, вивести значення усіх вершин рівня L, а також їх кількість N (якщо дерево не має вершин рівня L, то вивести 0)

Лістинг 9.5. – Код програми

```
#include <iostream>
```

```
using namespace std;
struct Node {
    double data;
    Node* left;
    Node* right;
};
Node* a(double value) {
    Node* newNode = new Node;
    newNode->data = value;
    newNode->left = nullptr;
    newNode->right = nullptr;
    return newNode;
}
Node* b(Node* root, double value) {
    if (root == nullptr) return a(value);
    if (value < root->data)
        root->left = b(root->left, value);
    else
        root->right = b(root->right, value);
    return root;
}
void c(Node* root, int currentLevel, int targetLevel, int& count) {
    if (root == nullptr) return;
    if (currentLevel == targetLevel) {
        cout << root->data << " ";
        count++;
    }
    else {
        c(root->left, currentLevel + 1, targetLevel, count);
        c(root->right, currentLevel + 1, targetLevel, count);
    }
}

int main() {
    Node* root = nullptr;
    double arr[] = { 10.5, 5.2, 15.3, 3.1, 7.4, 12.6, 18.9 };
    int n = sizeof(arr) / sizeof(arr[0]);
    for (int i = 0; i < n; i++) {
        root = b(root, arr[i]);
    }
    int L;
    cout << "Введіть рівень L: ";
    cin >> L;
    int count = 0;
    c(root, 0, L, count);
    if (count == 0) cout << "0";
    cout << "\nКількість вузлів рівня " << L << ": " << count <<
endl;
    cout << "Чавурська Єлизавета, П-31, ЛР9, AiCD";
    return 0;
}
```

A screenshot of a program window with a black background and yellow text. The text displays the results of a program execution, including a level number, a count of nodes, and a student's name and ID.

```
Введіть рівень L: 7  
0  
Кількість вузлів рівня 7: 0  
Чавурська Єлизавета, П-31, ЛР9, АіСД
```

Рисунок 9.5 – Вікно результатів програми

5. Зробити висновок про роботу.
6. Бути готовим до захисту J.

Висновок: на цій лабораторній роботі я набула практичних навичок побудови бінарних дерев та їх використання для розв'язування практичних задач.