Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної

техніки Кафедра інформатики та програмної

інженерії

Звіт

з лабораторної роботи № 5 з дисципліни
«Основи програмування-2. Методології програмування»
«Дерева»

Варіант 06

Виконав студент ІП-13 Вдовиченко Станіслав Юрійович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота №5
Дерева

Мета роботи: вивчити особливості організації і обробки дерев.

Завдання: Побудувати двійкове дерево пошуку з літер заданого рядка. Видалити з дерева літери, що зустрічаються більше одного разу. Вивести елементи дерева, що залишилися, при його постфіксному обході.

Код програми на C++

Tree.h

```
#include <iostream>
#include "string"
#include "cstdlib"
#include "vector"
using namespace std;

struct Branch{
    char Data;
    Branch* LeftBranch;
    Branch* RightBranch;
    Branch* Parent;
};

class Tree {
private:
    Branch *root;
    vector<char> duplicates;

    void insertElementRecursion(Branch*&, char, Branch*);
    void printTreeRecursion(Branch *&, int);
    int deleteElementsPrivate(char key);
    void postOrderRecursion(Branch*&);
    Branch** findBranch(Branch*&, char);
public:
    Tree();
    ~Tree();
    void addElement(char);
    void printTree();
    void deleteElements();
    void printPostOrder();

};
```

Tree.cpp

```
#include "Tree.h"
using namespace std;

Tree::Tree() {
    root = nullptr;
}
Tree::~Tree() {
    delete root;
}

void Tree::addElement(char str) {
    insertElementRecursion(root, str, nullptr);
}

void Tree::printTreeRecursion(Branch *& branch, int level) {
    char space = ' ';
    char under = '_';
    for (int i = 0; i < level; i++)
        cout << string(3, space) << "|";
    cout << string(2, under);
    if (branch != nullptr)
    {
        cout << branch->Data << "\n";
```

```
        printTreeRecursion(branch->RightBranch, level + 1);
        printTreeRecursion(branch->LeftBranch, level + 1);
    }
    else
        cout << "\n";
}

void Tree::printTree() {
    this->printTreeRecursion(root, 0);
}
void Tree::insertElementRecursion(Branch *& branch, char str, Branch*
parent) {
    if(branch == nullptr){
        branch = new Branch;
        branch->Data = str;
        branch->Parent = parent;
        branch->LeftBranch = nullptr;
        branch->RightBranch = nullptr;
    }
    else{
        if(str < branch->Data){
            insertElementRecursion(branch->LeftBranch, str, branch);
        }
        else if (str == branch->Data) {
            duplicates.push_back(str);
            return;
        }
        else{
            insertElementRecursion(branch->RightBranch, str, branch);
        }
    }
}
int Tree::deleteElementsPrivate(char key){

    Branch** current = findBranch(root, key);
    if (current == nullptr)
        return -1;

    if ((*current)->LeftBranch == nullptr && (*current)->RightBranch ==
nullptr) {
        Branch* tmp = *current;
        if ((*current)->Data < (*current)->Parent->Data)
            (*current)->Parent->LeftBranch = nullptr;
        else
            (*current)->Parent->RightBranch = nullptr;
        delete tmp;
        return 0;
    }

    if ((*current)->LeftBranch != nullptr && (*current)->RightBranch !=
nullptr) {
        auto leftmost = (*current)->RightBranch;

        while (leftmost && leftmost->LeftBranch != nullptr)
            leftmost = leftmost->LeftBranch;

        (*current)->Data = leftmost->Data;

        if (leftmost->RightBranch != nullptr) {
            leftmost->RightBranch->Parent = leftmost->Parent;
            auto tmp = leftmost->RightBranch;
```

```
            *leftmost = *leftmost->RightBranch;
            leftmost->Parent->LeftBranch = leftmost;
            delete tmp;
        } else {
            leftmost->Parent->RightBranch = nullptr;
            delete leftmost;
        }
        return 0;
    } else {
        if ((*current)->LeftBranch != nullptr) {
            Branch* tmp = *current;
            *current = (*current)->LeftBranch;
            (*current)->Parent = tmp->Parent;
            delete tmp;
        } else {
            auto tmp = *current;
            *current = (*current)->RightBranch;
            (*current)->Parent = tmp->Parent;
            delete tmp;
        }
        return 0;
    }
}

void Tree::postOrderRecursion(Branch*& branch) {
    if(branch == nullptr)
        return;
    postOrderRecursion(branch->LeftBranch);
    postOrderRecursion(branch->RightBranch);
    cout << branch->Data << " ";
}

void Tree::printPostOrder() {
    postOrderRecursion(root);
}

void Tree::deleteElements() {
    if (duplicates.empty()) return;
    for (char duplicate : duplicates) {
        deleteElementsPrivate(duplicate);
    }
}

Branch **Tree::findBranch(Branch *& branch, char key) {
    if (branch == nullptr)
        return nullptr;

    if (key == branch->Data)
        return &branch;

    if (key < branch->Data)
        return findBranch(branch->LeftBranch, key);
    else
        return findBranch(branch->RightBranch, key);
}
```

Main.cpp

```
#include "Tree.h"

int main() {
    Tree tree;
```

```
    string str;


    vector<char> array;
    cout << "Enter string: ";
    getline(cin, str);
    for (char & i : str) {
        if(i != ' ' && i != ',' && i != '.') {
            array.push_back(i);
        }
    }
    for (char i : array) {
        tree.addElement(i);
    }
    cout << "Tree: " << endl;
    tree.printTree();
    tree.deleteElements();
    cout << "Postorder (after deleting duplicates): " << endl;
    tree.printPostOrder();
    return 0;
}
```

Робота програми.

```
C:\Users\Stas\CLionProjects\Lab6\cmake-build-debug\Lab6.exe
Enter string:how are u doin today
 Tree:
__h
   |__o
   |   |__w
   |   |   |__y
   |   |   |   |__
   |   |   |   |__
   |   |   |__r
   |   |   |   |__u
   |   |   |   |   |__
   |   |   |   |   |__t
   |   |   |   |   |   |__
   |   |   |   |   |   |__
   |   |   |   |__
   |   |__i
   |   |   |__n
   |   |   |   |__
   |   |   |   |__
   |   |   |__
   |__a
   |   |__e
   |   |   |__
   |   |   |__d
   |   |   |   |__
   |   |   |   |__
   |   |__
Postorder (after deleting duplicates):
e n i t u y w r h
Process finished with exit code 0
```

Висновок: під час виконання даної лабораторної роботи я вивчив особливості організації і обробки дерев на основі мови С++.