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

**Кафедра ІПІ**

# Звіт

з лабораторної роботи № 6 з дисципліни
«Основи програмування 2. Модульне програмування»

**„ Дерева”**

Виконав *ІП-Левченко Владислав В’ячеславович*
(шифр, прізвище, ім'я, по батькові)

Перевірила *Вечерковська Анастасія Сергіївна*
(прізвище, ім'я, по батькові)

Київ 2022

## Код c++

### Tree.h

```
#pragma once

#include <iostream>
#include <iomanip>
#include <ctime>
#include <string>
#include <vector>

using namespace std;

class Node {
private:
	string data;
	Node* left;
	Node* right;
public:
	Node(string word);
	~Node();
	friend class Tree;
};
class Tree {
private:
	Node* root;
	void addNodeRecur(Node*& parent, string newWord);
	void printNodeOrder(Node*& parent, int level);
	int searchRecur(Node*& parent, string word, int
level);
public:
	Tree();
	~Tree();
	void addNode(string word);
	void printTree();
	int searchLevel(string word);
};
```

### Tree.cpp

```
#include "Tree.h"

Node::Node(string word) {
data = word;
```

```
left = NULL;
	right = NULL;
}
Node::~Node() {
	delete left;
	delete right;
}
void Tree::addNodeRecur(Node*& parent, string newWord) {
	if (parent == NULL)
		parent = new Node(newWord);
	else if (newWord < parent->data)
		addNodeRecur(parent->left, newWord);
	else if (newWord > parent->data)
		addNodeRecur(parent->right, newWord);
}
void Tree::printNodeOrder(Node*& parent, int level) {
	char space = ' ';
	char under = '_';
	for (int i = 0; i < level; i++)
		cout << string(3, space) << "|";
	cout << string(2, under);
	if (parent != NULL)
	{
		cout << parent->data << "\n";
		printNodeOrder(parent->right, level + 1);
		printNodeOrder(parent->left, level + 1);
	}
	else
		cout << "\n";
}
int Tree::searchRecur(Node*& parent, string word, int level) {
	if (parent == NULL)
		return -1;
	else if (parent->data == word)
		return level;
	else if (word < parent->data)
		return searchRecur(parent->left, word, level + 1);
	else
		return searchRecur(parent->right, word, level + 1);
}
Tree::Tree() {
	root = NULL;
}
Tree::~Tree() {
	delete root;
}
void Tree::addNode(string word) {
	addNodeRecur(root, word);
}
void Tree::printTree() {
	printNodeOrder(root, 0);
}
int Tree::searchLevel(string word) {
```

```
	int n = searchRecur(root, word, 0);
	return n;
}
```

## Function.h

```
#pragma once

#include "Tree.h"
#include <iostream>
#include <vector>
#include <string>
using namespace std;

vector <string> inputArr();
void printArr(vector <string>);
vector <string> enterWords();
string enterWordtoSearch();
bool isNumber(const string&);
void addToTree(vector <string>, Tree&);
void findWordnLevel(Tree&);
```

## Function.cpp

```
#include "Function.h"
#include "Tree.h"

vector <string> inputArr()
{
	vector <string> arr;
	arr = enterWords();
	return arr;
}
void printArr(vector <string> arr)
{
	for (int i = 0; i < arr.size(); i++)
		cout << arr[i] << " ";
	cout << "\n";
}
vector <string> enterWords()
{
	vector <string> arr;
	cout << "Enter words:\n";
	string str;
	for (int i = 1; ; i++)
	{
		cout << i << ") ";
		getline(cin, str);
		if (str == "") { break; }
```

```
			arr.push_back(str);
		}
		return arr;
}
string enterWordtoSearch()
{
		string word;
		cout << "\nEnter word to search: ";
		cin >> word;
		while (word.length() < 1)
		{
			cout << "Enter correct word:";
			cin >> word;
		}

		return word;
}

bool isNumber(const std::string& word) {
		for (char i : word) {
			if (!isdigit(i)) return false;
		}
		return true;
}

void addToTree(vector<string> arr, Tree &tree)
{
		for (int i = 0; i < arr.size(); i++)
			tree.addNode(arr[i]);
}

void findWordnLevel(Tree &tree)
{
		string word = enterWordtoSearch();
		int level = tree.searchLevel(word);
		if (level != -1)
			cout << "\nlevel of word '" << word << "' = " << level
+ 1 << "\n";
		else
			cout << "\nThere is no word '" << word << "'\n";
}
```

## main

```
#include "Tree.h"
#include "Function.h"

int main()
{
		vector <string> arr = inputArr();
```

```
        cout << "\nEntered words:\n";
        printArr(arr);

        Tree tree;
        addToTree(arr, tree);

        cout << "\nThe tree:\n";
        tree.printTree();
        findWordnLevel(tree);

        cin.ignore(2);
}
```

## Випробування:

```
C:\Users\Admин\source\repos\lab6 c++\Debug\lab6 c++.exe
To stop writing lines to the tree - press ENTER

Enter words:
1) book
2) alg
3) dimension
4) troll
5) aleksrozik
6)

Entered words:
book alg dimension troll aleksrozik

The tree:
__book
   |__dimension
   |    |__troll
   |    |    |__
   |    |    |__
   |    |__
   |__alg
   |    |__
   |    |__aleksrozik
   |    |    |__
   |    |    |__

Enter word to search: alg

level of word 'alg' = 2
```

Висновки: під час виконання даної лабораторної роботи я навчився особливостям організації і обробки дерев. Я створив бінарне дерево, я у яке записав довільне число слів і знайшов рядок введеного слова.