Додаток 1

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. Модульне програмування»

«Дерева»

Варіант 15

Виконав студент ІП-13 Карамян Вартан Суренович
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

Лабораторна робота 5

Дерева

Варіант 15

15. Текстовий файл містить програму мовою C/C++. Для збереження ідентифікаторів програми використати структуру типу дерева, елементами якого є ідентифікатори. Номер рядка, в якому оголошений ідентифікатор, визначає рівень дерева. Ліва гілка дерева визначає змінні, права гілка - константи.

Код програми на мові C++:

lab5 C++.cpp

```
#include "Functions.h"

int main()
{
	Tree tree;
	string filename = getFileName();

	printFile(filename);
	readFile(filename, tree);

	tree.show();
}
```

Tree.h

```
#pragma once
#include <string>
#include <iostream>

using namespace std;

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

class Tree
{
	Node* root;
	Node* insertRecursion(Node*, string, int, string);
	void showRecursion(const string&, const Node*, bool);
public:
	Tree() :root(NULL) {};
	~Tree();
```

```
        void insert(string, int, string);
        void show();
};
```

Tree.cpp

```
#include "Tree.h"

Node::~Node()
{
        delete left;
        delete right;
}

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

Node* Tree::insertRecursion(Node* temp, string value, int key, string branch)
{
        if (temp == NULL)
        {
                return new Node(value, key);
        }
        if (branch == "left")
        {
                temp->left = insertRecursion(temp->left, value, key, branch);
        }
        else if (branch == "right")
        {
                temp->right = insertRecursion(temp->right, value, key, branch);
        }
        return temp;
}

void Tree::insert(string value, int key, string branch)
{
        if (root == NULL)
        {
                root = insertRecursion(root, value, key, branch);
        }
        else
        {
                insertRecursion(root, value, key, branch);
        }
}

void Tree::showRecursion(const string& prefix, const Node* temp, bool isRight)
{
        if (temp != NULL)
        {
                cout << prefix;
                cout << (isRight ? "|--" : "\\--");
                cout << temp->data << "(" << temp->key << ")" << endl;
                showRecursion(prefix + (isRight ? "|       " : "        "), temp->right, true);
                showRecursion(prefix + (isRight ? "|       " : "        "), temp->left, false);
        }
}

void Tree::show()
{
        cout << "\nTree:\n";
        showRecursion("", root, false);
        cout << "\n\n";
```

```
}
```

Functions.h

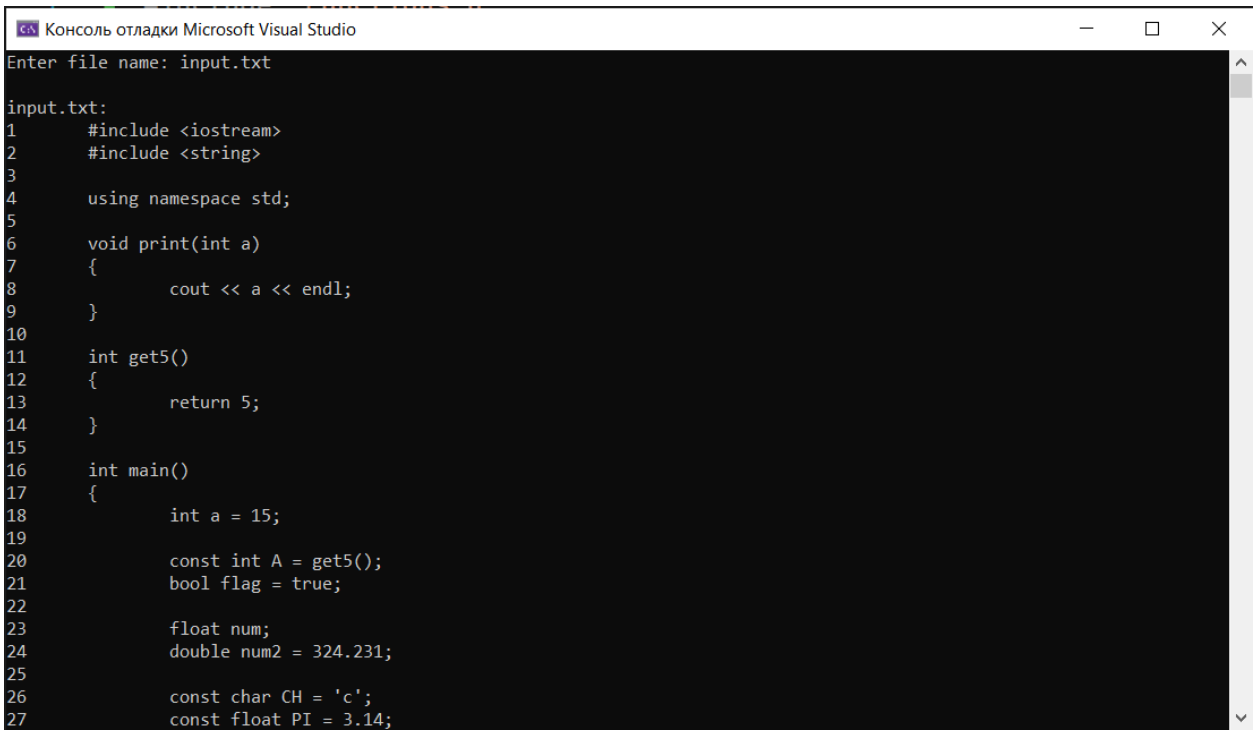
```
#pragma once
#include <fstream>
#include <vector>
#include "Tree.h"

string getFileName();

void printFile(string);

void readFile(string, Tree&);

void findIdentifier(Tree& , string , int , string );
```

Functions.cpp

```
#include "Functions.h"

string getFileName()
{
	string filename;
	cout << "Enter file name: ";
	cin >> filename;
	return filename;
}

void printFile(string filename)
{
	ifstream fin(filename);

	if (!fin.is_open())
	{
		cout << "Error. File is not found.\n";
	}
	else
	{
		cout<<endl << filename << ":\n";
		string line;
		int counter = 0;
		while (!fin.eof())
		{
			getline(fin, line);
			counter++;
			cout << counter << "\t" << line << endl;
		}
		cout << endl;
	}

	fin.close();
}

void readFile(string filename, Tree& tree)
{
	ifstream fin(filename);

	if (!fin.is_open())
	{
		cout << "Error. File is not found.\n";
	}
	else
	{
```

```
                string line;
                int counter = 0;

                while (!fin.eof())
                {
                        getline(fin, line);
                        counter++;
                        vector<string> types{ "int ","float ","double ","char ","bool " };
                        bool condition = false;
                        string type = "";

                        for (size_t i = 0; i < types.size(); i++)
                        {
                                if (line.find(types[i]) < line.length())
                                {
                                        condition = true;
                                        type = types[i];
                                        break;
                                }
                        }

                        if (condition && line.find(";") < line.length())
                        {
                                findIdentifier(tree, line, counter, type);
                        }
                }

        }

        fin.close();
}

void findIdentifier(Tree& tree, string line, int counter, string type)
{
        int startIndex = line.find(type) + type.length();
        int idLength = line.length() - startIndex - 1;

        if (line.find("=") < line.length())
                idLength -= (line.length() - line.find("="));

        string temp = line.substr(startIndex, idLength);

        if (line.find("const ") < line.length())
        {
                tree.insert(temp, counter, "right");
        }
        else
        {
                tree.insert(temp, counter, "left");
        }
}
```

Результати роботи:



Висновок:

В результаті виконання лабораторної роботи ми вивчили та закріпили на практиці використання такої структури даних як дерево.