Міністерство освіти і науки України

Івано-Франківський національний технічний університет нафти й газу

Кафедра комп’ютерних систем і мереж

Лабораторна робота №5

на тему:

**“**ОРГАНІЗАЦІЯ ТАБЛИЦЬ ІДЕНТИФІКАТОРІВ **”**

Виконав:

студент групи КІ-21-1К

Шиманський Б.В.

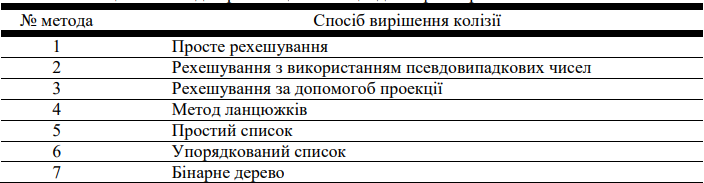
Перевірив: викладач

Гарасимів В. М.

**Мета:** вивчити основні методи\ організації таблиць ідентифікаторів, отримати уявлення про переваги і недоліки, властиві різним методам організації таблиць ідентифікаторів.

**Завдання**

Для виконання лабораторної роботи потрібно написати програму, яка отримує на вході набір ідентифікаторів, організовує таблиці ідентифікаторів за допомогою заданих методів, дозволяє здійснити багатократний пошук довільного ідентифікатора в таблицях і порівняти ефективність методів організації таблиць. Список ідентифікаторів вважати заданим у вигляді текстового файлу. Довжина ідентифікаторів обмежена 32 символами.



**Завдання**



**Хід роботи**

**Код програми**

#include <iostream>

#include <fstream>

#include <vector>

#include <string>

#include <chrono>

using namespace std;

// Вузол бінарного дерева

struct TreeNode {

string value;

TreeNode\* left;

TreeNode\* right;

TreeNode(const string& val)

: value(val), left(nullptr), right(nullptr) {}

};

// Клас бінарного дерева

class BinaryTree {

public:

BinaryTree() : root(nullptr) {}

// Вставка значення в бінарне дерево

void insert(const string& value) {

root = insertRecursive(root, value);

}

// Пошук значення в бінарному дереві

bool search(const string& value) const {

return searchRecursive(root, value);

}

private:

TreeNode\* root;

// Рекурсивна функція для вставки значення в бінарне дерево

TreeNode\* insertRecursive(TreeNode\* node, const string& value) {

if (node == nullptr) {

return new TreeNode(value);

}

if (value < node->value) {

node->left = insertRecursive(node->left, value);

}

else if (value > node->value) {

node->right = insertRecursive(node->right, value);

}

return node;

}

// Рекурсивна функція для пошуку значення в бінарному дереві

bool searchRecursive(const TreeNode\* node, const string& value) const {

if (node == nullptr) {

return false;

}

if (value == node->value) {

return true;

}

else if (value < node->value) {

return searchRecursive(node->left, value);

}

else {

return searchRecursive(node->right, value);

}

}

};

// Клас простого списку

class SimpleList {

public:

SimpleList() {}

// Вставка значення в список

void insert(const string& value) {

items.push\_back(value);

}

// Пошук значення в списку

bool search(const string& value) const {

for (const auto& item : items) {

if (item == value) {

return true;

}

}

return false;

}

private:

vector<string> items;

};

// Функція для зчитування списку ідентифікаторів з файлу

vector<string> readIdentifiers(const string& filename) {

vector<string> identifiers;

ifstream infile(filename);

string line;

while (getline(infile, line)) {

identifiers.push\_back(line);

}

return identifiers;

}

int main() {

// Зчитування списку ідентифікаторів з файлу

vector<string> identifiers = readIdentifiers("identifiers.txt");

// Створення бінарного дерева та простого списку

BinaryTree binaryTree;

SimpleList simpleList;

// Вставка ідентифікаторів в бінарне дерево та простий список

for (const auto& identifier : identifiers) {

binaryTree.insert(identifier);

simpleList.insert(identifier);

}

// Пошук довільного ідентифікатора в бінарному дереві

string searchValue = "identifier12";

auto start = chrono::high\_resolution\_clock::now();

bool found = binaryTree.search(searchValue);

auto end = chrono::high\_resolution\_clock::now();

cout << "Binary tree search for '" << searchValue << "' took "

<< chrono::duration\_cast<chrono::microseconds>(end - start).count()

<< " microseconds. Found: " << (found ? "yes" : "no") << endl;

// Пошук довільного ідентифікатора в простому списку

start = chrono::high\_resolution\_clock::now();

found = simpleList.search(searchValue);

end = chrono::high\_resolution\_clock::now();

cout << "Simple list search for '" << searchValue << "' took "

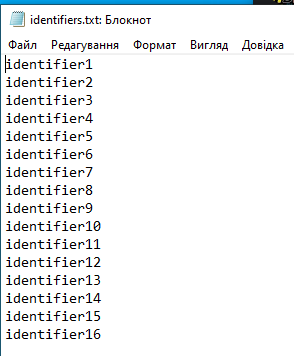
<< chrono::duration\_cast<chrono::microseconds>(end - start).count()

<< " microseconds. Found: " << (found ? "yes" : "no") << endl;

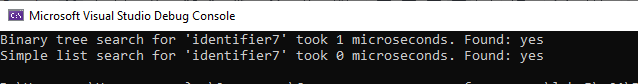
return 0;

}

**Вміст файлу**



**Результат виконання програми**



GitHub репозиторій з виконаними лабораторними роботами:

https://github.com/Rocker7000/labs

**Висновки:** Навчився формувати та обробляти основні компонентні структури трансляторів (компіляторів): графи, дерева, списки, стьоки і черги при.