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 № 6 з дисципліни

«Основи програмування-2.

Методології програмування»

«Дерева»

Варіант 31

Виконав студент ІП-11 Трикош Іван Володимирович

Перевірила Вітковська Ірина Іванівна

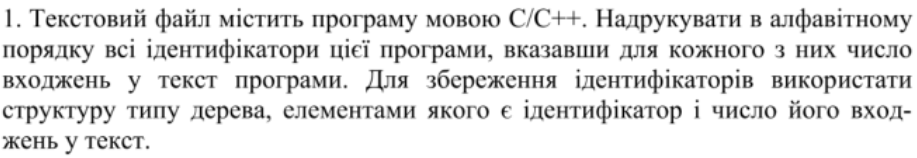
Київ 2022

**Лабораторна робота №6**

**Дерева**

**Мета –** вивчити особливості організації та обробки дерев.

Варіант 31.



**Постановка задачі.** Спочатку вводимо зчитуємо файл та виділяємо і ньому ідентифікатори, записуючи їх у масив. Далі створюємо дерево, коренем якого є перший ідентифікатор. Потім проходимо по масиву ідентифікаторів і додаємо їх у дерево або збільшуємо кількість входжень. Після цього виводимо центровано дерево і видаляємо його.

**Код та результат програми на С++:**

**Func.h**

#pragma once

#include <iostream>

#include <string>

#include <vector>

#include <fstream>

using namespace std;

class Tree

{

private:

string name; // Ім'я ідентифікатора

int number; // Кількість входжень

Tree\* left; // Права гілка дерева

Tree\* right; // Ліва гілка дерева

public:

Tree(string); // Створюємо дерево з одним ключем

void Output\_tree(Tree\*); // Виводимо дерево

void insert\_name(Tree\*, string); // Всатвляємо елемент у дерево

~Tree(); // Видаляємо дерево

};

void readfile(vector <string>&, string&); // Зчитуємо ідентифікатори

**Laba-C++.cpp**

#include "Func.h"

int main()

{

vector <string> identificators; // Масив ідентифікаторів

string filename = "file.txt"; // Ім'я файлу

readfile(identificators, filename); // Зчитуємо ідентифікатори

if (identificators.size() == 0) // Перевіряємо, чи є ідентифікатори

{

cout << "There are no identificators" << endl;

}

Tree tree(identificators[0]); // Стоворюємо дерево з першим іденфікатором

for (int i = 1; i < identificators.size(); i++) // Додаємо ідентифікатори у дерево

{

tree.insert\_name(&tree, identificators[i]);

}

tree.Output\_tree(&tree); // Виводимо дерево

return 0;

}

**Func.cpp**

#include "Func.h"

Tree::Tree(string name) // Ініціалізуємо корінь дерева заданим та початковими значеннями

{

this->name = name;

this->number = 1;

this->left = nullptr;

this->right = nullptr;

}

void Tree::insert\_name(Tree\* node, string name) // Вставляємо елемент у дерево

{

if (name < node->name) // Якщо елемент стоїть першим за алфавітом, ніж поточний ключ

{

if (node->left == nullptr) // Якщо лівої гілки нема

{

node->left = new Tree(name); // Створюємо ліву гілку

}

else

{

insert\_name(node->left, name); // Інакше рекурсивно викликаємо функцію

}

}

else if (name > node->name) // Якщо поточний ключ стоїть першим за алфавітом, ніж елемент

{

if (node->right == nullptr) // Якщо правої гілки нема

{

node->right = new Tree(name); // Створюємо праву гілку

}

else

{

insert\_name(node->right, name); // Інакше рекурсивно викликаємо функцію

}

}

else

{

node->number++; // Інакше збільшуємо кількість входжень у текст програми

}

}

void Tree::Output\_tree(Tree\* node) // Виводимо центровано дерево

{

if (node->left != nullptr)

{

Output\_tree(node->left);

}

cout << node->name << " - " << node->number << endl;

if (node->right != nullptr)

{

Output\_tree(node->right);

}

}

Tree::~Tree() // Видаляємо дерево

{

delete this->left;

delete this->right;

}

void readfile(vector <string>& identificators, string& filename)

{

ifstream file(filename); // Відкриваємо для читання файл

string temp; // Поточний рядок

const int n = 84; // Розмір масиву не ідентифікаторів

string notidentificators[n] = {"asm", "auto", "bool", "break", "case", "catch", "char", "class", "const", "const\_class", "continue",

"default", "delete", "double", "do", "dinamic\_cast", "else", "enum", "explicit", "export", "extern", "false", "float", "for", "friend",

"goto", "if", "inline", "int", "long", "mutable", "namespace", "new", "operator", "private", "protected", "public", "register",

"reinterpret\_cast", "return", "short", "unsigned", "signed", "sizeof", "static", "static\_cast", "struct", "switch", "template", "this", "throw",

"true", "try", "typedef", "typeid", "typename", "union", "using", "virtual", "void", "volatile", "wchar\_t", "while", "{",

"}", "[", "]", "&", "+", "-", "\*", "/", "->", "%", ">>", "<<", "++", "--", "#", "(", ")", ";", ",", "="}; // Масив не ідентифікаторів

while (!file.eof()) // Проходимо до кінця файлу

{

getline(file, temp); // Зчитуємо поточний рядок

if (temp[0] != '#') // Перевіряємо рядок на символ "#" і за потреби пропускаємо його

{

int pos; // Позиція елемента

pos = temp.find('\''); // Видаляємо символ "'" і все, що він означує

while (pos != -1)

{

temp.erase(pos, 3);

if (pos == 0 && temp.size() != 0)

{

temp.erase(pos, 1);

}

else if (pos != 0 && pos == temp.size())

{

temp.erase(pos - 1, 1);

}

else if (pos != 0)

{

temp.erase(pos - 1, 1);

}

pos = temp.find('\'');

}

pos = temp.find('\"'); // Видаляємо символ '"' і все, що він означує

while (pos != -1)

{

int pos2 = temp.find('\"', pos + 1);

temp.erase(pos, pos2 - pos + 1);

if (pos == 0 && temp.size() != 0)

{

temp.erase(pos, 1);

}

else if (pos != 0 && pos == temp.size())

{

temp.erase(pos - 1, 1);

}

else if (pos != 0)

{

temp.erase(pos - 1, 1);

}

pos = temp.find('\"');

}

for (int i = 0; i < n; i++) // Видаляємо всі не ідентифікатори з рядка

{

int pos = temp.find(notidentificators[i]);

while (pos != -1)

{

temp.erase(pos, notidentificators[i].size());

if (pos == 0 && temp.size() != 0)

{

temp.erase(pos, 1);

}

else if (pos != 0 && pos == temp.size())

{

temp.erase(pos - 1, 1);

}

else if (pos != 0)

{

temp.erase(pos - 1, 1);

}

pos = temp.find(notidentificators[i]);

}

}

while (temp.size() != 0) // Записуємо ідентифікатори у вектор

{

int pos1 = temp.find(' ');

int pos2 = temp.find(' ', pos1 + 1);

if (pos1 != -1 && pos2 != -1)

{

string t = temp.substr(pos1 + 1, pos2 - pos1 - 1);

bool flag = false;

for (int j = 0; j < t.size(); j++)

{

if (isalpha(t[j]))

{

flag = true;

}

}

if (flag)

{

identificators.push\_back(t);

}

temp.erase(pos1, pos2 - pos1);

}

else if (pos1 != -1)

{

string t = temp.substr(pos1 + 1, temp.size() - pos1 - 1);

bool flag = false;

for (int j = 0; j < t.size(); j++)

{

if (isalpha(t[j]))

{

flag = true;

}

}

if (flag)

{

identificators.push\_back(t);

}

temp.erase(pos1);

}

else

{

bool flag = false;

for (int j = 0; j < temp.size(); j++)

{

if (isalpha(temp[j]))

{

flag = true;

}

}

int k = 0;

while (temp[k] == '\t')

{

temp.erase(k, 1);

k++;

}

k = temp.size() - 1;

if (flag)

{

identificators.push\_back(temp);

}

temp.clear();

}

}

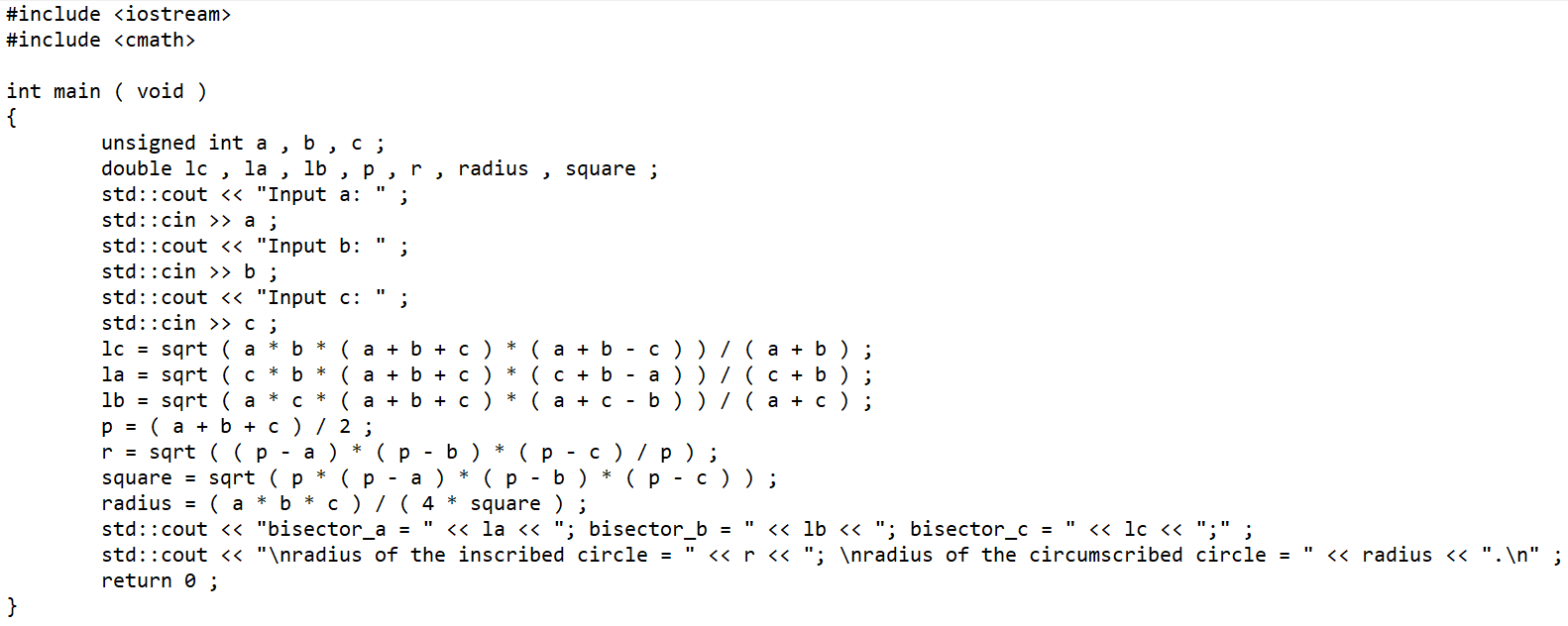
}

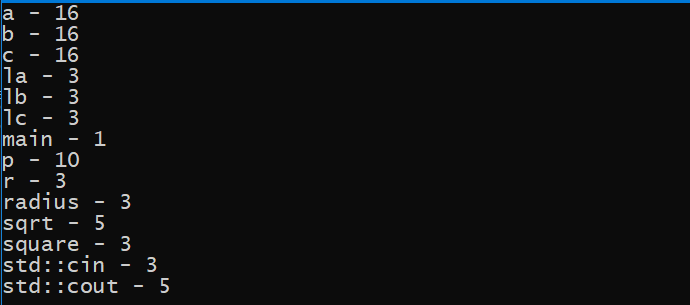
}

file.close();

}

Файл file.txt



****

**Висновок –** я вивчив особливості організації та обробки дерев на мові С++; удосконалив свої навички роботи з деревовидними структурами; створив програму, яка зчитує ідентифікатори з файлу, записує їх у масив, далі створює дерево і записує туди елементи в алфавітному порядку, виводить центровано дерево.