# 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»

Варіант 13

Виконав студент Калашніков Андрій Євгенович

(шифр, прізвище, ім'я, по батькові)

Перевірив

( прізвище, ім'я, по батькові)

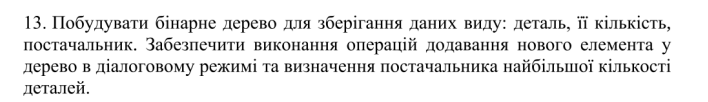
Київ 2021

**Лабораторна робота №5**

# **Дерева**

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

**Індивідуальне завдання:**



**Постановка задачі**

Розробимо бінарне дерево для зберігання деталей, їх кількості та постачальника. Реалізуємо функції. Додавання нового елементу у дерево в діалоговому режимі. Визначимо постачальника найбільшої кількості деталей.

**Код на С++ :**

**Header.h :**

#pragma once

#include <iostream>

#include<string>

#include<vector>

#include <algorithm>

using namespace std;

struct Detail {

string detail;

int amount;

string provider;

Detail\* left;

Detail\* right;

void print\_detail();

Detail(){

detail = "";

amount = 0;

provider = "";

left = 0;

right = 0;

}

};

void add\_detail(Detail\* &newOne, Detail\* &current);

void print\_tree(Detail\*& current);

void biggest\_provider(Detail\*& current, vector<int>&detailamount,int& i, vector<string>& providers);

**Header.cpp**

#include "Header.h"

void Detail::print\_detail() {

cout << endl << "Name: " << detail << " Amount: " << amount << " Provider: " << provider;

}

void add\_detail(Detail\* &newOne, Detail\* &current) {

Detail\* p = new Detail;

p = newOne;

if (current) {

if (p->detail > current->detail) {

add\_detail(newOne, current->right);

}

else if (p->detail < current->detail) {

add\_detail(newOne, current->left);

}

}

else {

current = newOne;

}

}

void print\_tree(Detail\*& current) {

Detail\* p = new Detail;

p = current;

if (p) {

p->print\_detail();

print\_tree(p->left);

print\_tree(p->right);

}

else {

}

}

void biggest\_provider(Detail\*& current, vector<int>&detailamount, int &i, vector<string>&providers) {

Detail\* p = new Detail;

p = current;

if (p) {

providers[i]=p->provider;

detailamount[i] = p->amount;

i++;

biggest\_provider(p->left,detailamount,i,providers);

biggest\_provider(p->right, detailamount, i, providers);

}

else {

}

}

**Source.cpp:**

#include "Header.h"

int main() {

char a;

int n=0;

Detail\* root = NULL;

Detail\* current = new Detail;

current = root;

cout << "Do you want to add new detail?(Y/N): ";

cin >> a;

while (a != 'n' && a != 'N') {

Detail\* newOne = new Detail;

cin.ignore();

cout << endl << "Enter detail name: ";

getline(cin, newOne->detail);

cout << endl << "Enter amount: ";

cin >> newOne->amount;

cin.ignore();

cout << endl << "Enter Provider: ";

getline(cin, newOne->provider);

if (n == 0) {

root = newOne;

current = newOne;

}

else {

add\_detail(newOne, root);

}

n++;

cout <<endl<< "Do you want to add new detail?(Y/N): ";

cin >> a;

}

print\_tree(root);

vector<int>detailamount(n);

int i=0;

vector<string>providers(n);

biggest\_provider(root,detailamount,i,providers);

auto iter = detailamount.cbegin();

auto iter2 = providers.cbegin();

for (int i = 0; i < detailamount.size(); i++) {

for (int j = i+1; j < detailamount.size(); j++) {

if (providers[i] == providers[j]) {

detailamount[i] += detailamount[j];

detailamount.erase(iter + j);

providers.erase(iter2 + j);

}

}

}

int temp;

int temp\_index;

int max,max\_index;

max = \*max\_element(detailamount.begin(), detailamount.end());

for (int i = 0; i < detailamount.size(); i++) {

if (detailamount[i] == max) {

max\_index = i;

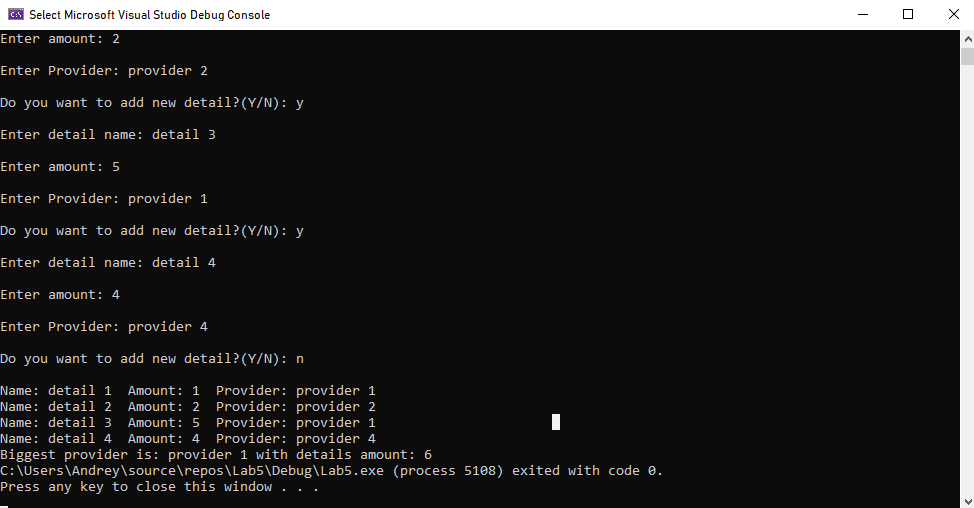
}

}

cout <<endl<< "Biggest provider is: "<<providers[max\_index]<<" with details amount: "<<max;

}

**Скріншот результатів програми на C++ :**

****

**Посилання на GitHub:**

**https://github.com/Andrey7904/KPI/tree/main/OP-2/Lab5**

**Висновок:**

Під час лабораторної роботи я вивчив механізми створення і використання бінарних дерев. Оскільки програма правильно створює дерево з вказаними даними, правильно додає новий елемент у дерево в діалоговому режимі, визначає постачальника найбільшої кількості деталей, то вона працює правильно.