# 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»

Варіант 13

Виконав студент Котков Тимур Максимович

(шифр, прізвище, ім'я, по батькові)

Перевірив Вітковська Ірина Іванівна

( прізвище, ім'я, по батькові)

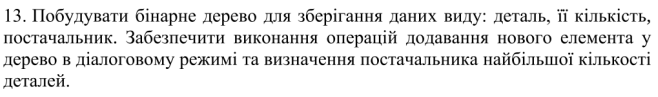
Київ 2022

**Лабораторна робота №6**

**Дерева**

**Мета –** визначити механізми створення класів та об’єктів.

**Індивідуальне завдання:**

****

**Постановка задачі**

Розробимо структуру Tree з полями, що приймають як параметри назву деталі, її кількість та її провайдера. Реалізуємо функцію визначення постачальника найбільшої кількості деталей.

**Код на С++ :**

**Lab\_6.cpp:**

#include "Tree.h"

#include <iostream>

#include <string>

using namespace std;

int main() {

int n = 0;

char enterNewDetail;

Detail\* root = NULL;

Detail\* current = new Detail;

current = root;

cout << "Do you want to add new detail?(y/n): "; cin >> enterNewDetail;

while (tolower(enterNewDetail) != 'n') {

Detail\* nItem = new Detail;

cin.ignore();

cout << endl << "Detail name: ";

getline(cin, nItem->detail);

cout << endl << "Amount: ";

cin >> nItem->amount;

cin.ignore();

cout << endl << "Provider: ";

getline(cin, nItem->provider);

if (n == 0) {

root = nItem;

current = nItem;

}

else {

add\_detail(nItem, root);

}

n++;

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

cin >> enterNewDetail;

}

output\_tree(root);

int maxNumDetails;

string bestProvider = biggestProvider(maxNumDetails, root, n);

cout << endl << "Provider with a most amount of details is: " << bestProvider << " with details amount: " << maxNumDetails;

}

**Tree.cpp:**

#include "Tree.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 output\_tree(Detail\*& current) {

Detail\* p = new Detail;

p = current;

if (p) {

p->print\_detail();

output\_tree(p->left);

output\_tree(p->right);

}

}

void getInfoProviders(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++;

getInfoProviders(p->left,detailamount,i,providers);

getInfoProviders(p->right, detailamount, i, providers);

}

}

string biggestProvider(int& max, Detail\* root, int n) {

vector<int>detailamount(n);

int i = 0;

vector<string>providers(n);

getInfoProviders(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\_index;

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

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

if (detailamount[i] == max) {

max\_index = i;

}

}

return providers[max\_index];

}

**Tree.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;

}

};

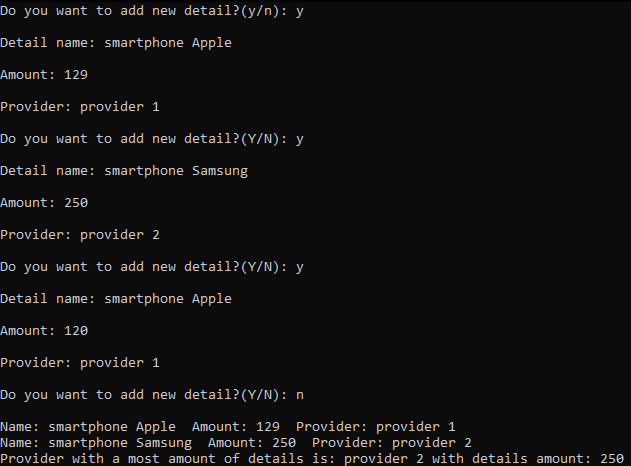
void add\_detail(Detail\*& newOne, Detail\*& current);

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

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

string biggestProvider(int& max, Detail\* root, int n);

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

****

**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.