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»

Варіант 25

Виконав студент ІП-14 Радзівіло Валерія Артемівна

(шифр, прізвище, ім'я, по батькові)

Перевірив

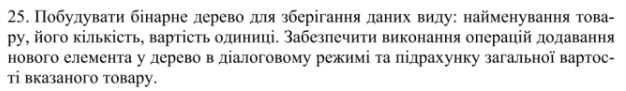
( прізвище, ім'я, по батькові)

Київ 2022

**Лабораторна робота №5**

**Тема роботи:** Дерева

**Мета роботи:** Вивчити особливості організації і обробки дерев.



**Вирішення задачі С++:**

**main:**

#include "header.cpp"  
  
int main() {  
 // flag to add something to the program  
 int again = 0;  
 // amount of products  
 int am\_prods = 0;  
 // array of products  
 vector<Product> prods;  
 do {  
 int to\_end;  
 do {  
 Product good;  
 // get data from user  
 good.setProduct();  
 // cout total cost for each position (used later for displaying)  
 good.countTotal();  
 // flag to check if the same product with the same price was before  
 int check = check\_same(prods, good);  
 if (check == -1) // if it wasn't before  
 prods.push\_back(good);  
 else {  
 prods[check].amount += good.amount;  
 prods[check].total = prods[check].price \* prods[check].amount;// re-count total  
 cout<<"You've already entered this product with the same price. The amount is added."<< endl;  
 am\_prods--; // this doesn't add any position to list  
 }  
 // user can input as much products as they want  
 cout << "Print 0 to end input: ";  
 cin >> to\_end;  
 cin.ignore();  
 am\_prods++;  
 } while (to\_end != 0);  
 // sort an array by names  
 sort\_by\_name(prods);  
 cout << endl <<"Your input: " << endl;  
 output\_arr(prods); // output array  
 // create trees  
 cout <<endl << "Trees: " << endl;  
 make\_trees(prods, am\_prods);  
 // get total of a product  
 cout<<"Which product's total cost do you want?"<< endl;  
 string total\_prod;  
 getline(cin, total\_prod);  
 // check if the total exists  
 int total\_cost = find\_total(prods, total\_prod);  
 if(total\_cost!=0)  
 cout <<"The total cost is "<< total\_cost << endl;  
 else  
 cout <<"You haven't entered this product!"<< endl;  
 // user can add more products to the same program  
 cout << "Do you want to add something?(print '1' if yes) ";  
 cin >> again;  
 cin.ignore();  
 }while(again==1);  
}

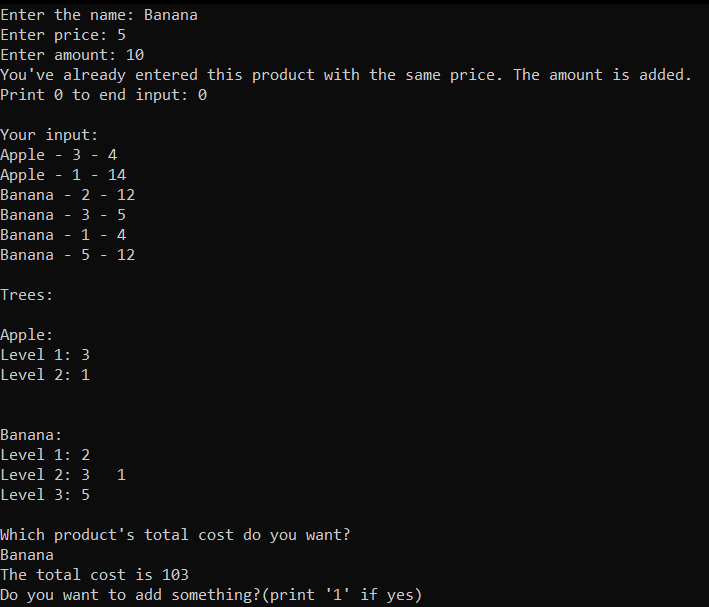
**header.h:**

#ifndef INC\_6LAB\_HEADER\_H  
#define INC\_6LAB\_HEADER\_H  
#include <iostream>  
#include <vector>  
#include <algorithm>  
#include <bits/stdc++.h>  
using namespace std;  
class Product {  
public:  
 string name;  
 int price;  
 int amount;  
 int total;  
 void setProduct();  
 Product();  
 Product(string name, int price, int amount);  
 void output\_product();  
 void countTotal();  
};  
struct TreeNode {  
 int val;  
 TreeNode \*left;  
 TreeNode \*right;  
 TreeNode(int x) : val(x), left(NULL), right(NULL) {}  
};  
  
void createTree(TreeNode\* node, int i, vector<string>& arr); // used in levelOrderCreateTree, fills a tree  
TreeNode\* levelOrderCreateTree(vector<string>& arr); // used in make\_trees to pass 1 value to pass head  
void printCurrentLevel(TreeNode\* root, int level);// Prints current level of tree  
int height(TreeNode\* node); // Compute the "height" of a tree - number of levels  
void sort\_by\_name(vector <Product> &prods); // sort an array by name  
void output\_arr(vector <Product> prods); // print array  
void one\_num\_tree(Product pr); // if a tree has only 1 value  
void printLevelOrder(TreeNode\* root); // function to print in level order  
void make\_trees(vector<Product>prods, int am\_prods); // create and fill all trees  
int check\_same(vector<Product>prods, Product good); // check if object is already in a list  
int find\_total(vector<Product>prods, string good); // count total for the product  
  
#endif //INC\_6LAB\_HEADER\_H

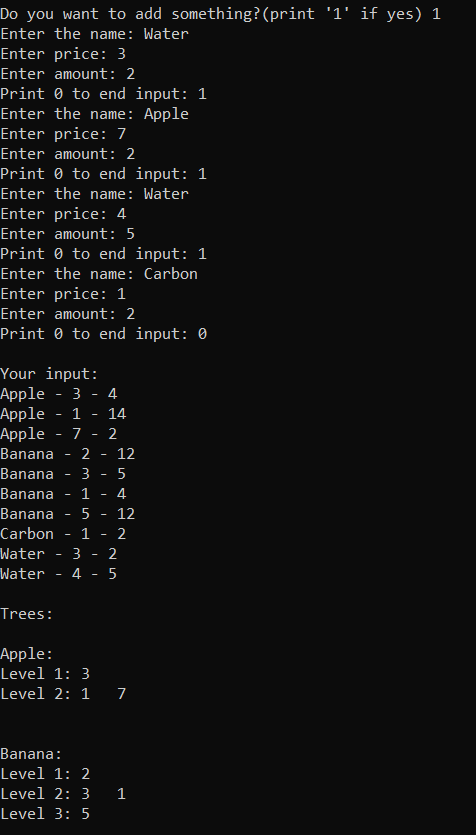
**header.cpp:**

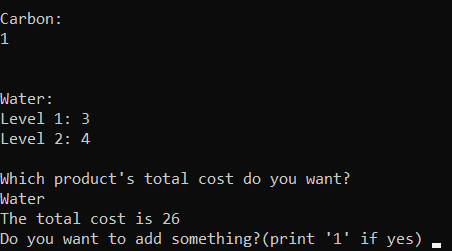
#include "header.h"  
  
Product::Product() {  
 name = " ";  
 price = 0;  
 amount = 0;  
 total = 0;  
}  
  
Product::Product(string name, int price, int amount) {  
 this->name = name;  
 this->price = price;  
 this->amount = amount;  
 this->total = price \* amount;  
}  
  
void Product::setProduct() {  
 cout << "Enter the name: ";  
 getline(cin, name);  
 cout << "Enter price: ";  
 cin >> price;  
 cout <<"Enter amount: ";  
 cin >> amount;  
}  
void Product::output\_product() {  
 cout << name << " - "<< price <<" - "<< amount<< endl;  
}  
  
void Product::countTotal() {  
 total = price\*amount;  
}  
  
void sort\_by\_name(vector <Product> &prods)  
{  
 sort(prods.begin(), prods.end(), [](const Product& lhs, const Product& rhs) {  
 return lhs.name < rhs.name;  
 });  
}  
  
void output\_arr(vector <Product> prods)  
{  
 for (auto & prod : prods)  
 prod.output\_product();  
}  
  
void one\_num\_tree(Product pr) {  
 cout << pr.name<< ":"<< endl;  
 cout <<pr.price<< endl;  
}  
  
void printLevelOrder(TreeNode\* root) // print tree in a level order  
{  
 int h = height(root);  
 int i;  
 for (i = 1; i <= h; i++) {  
 cout <<"Level "<<i<<": ";  
 printCurrentLevel(root, i);  
 cout << endl;  
 }  
}  
  
// Prints current level of tree  
void printCurrentLevel(TreeNode\* root, int level)  
{  
 if (root == NULL)  
 return;  
 if (level == 1)  
 cout << root->val << " ";  
 else if (level > 1) {  
 printCurrentLevel(root->left, level - 1);  
 printCurrentLevel(root->right, level - 1);  
 }  
}  
  
// Compute the "height" of a tree - number of levels  
int height(TreeNode\* tree)  
{  
 if (tree == NULL)  
 return 0;  
 else {  
 // compute the height of each subtree(left/right)  
 int lheight = height(tree->left);  
 int rheight = height(tree->right);  
  
 // use the biggest  
 if (lheight > rheight) {  
 return (lheight + 1);  
 }  
 else {  
 return (rheight + 1);  
 }  
 }  
}  
void createTree(TreeNode\* tree, int i, vector<string>& arr) { // used in levelOrderCreateTree  
 if (tree != NULL) {  
 if (2 \* i + 1 < arr.size()) {  
 if (arr[2 \* i + 1] == "null")  
 tree->left = NULL;  
 else  
 tree->left = new TreeNode(stoi(arr[2 \* i + 1]));  
 createTree(tree->left, 2 \* i + 1, arr);  
 }  
  
 if (2 \* i + 2 < arr.size()) {  
 if (arr[2 \* i + 2] == "null")  
 tree->right = NULL;  
 else  
 tree->right = new TreeNode(stoi(arr[2 \* i + 2]));  
 createTree(tree->right, 2 \* i + 2, arr);  
 }  
 }  
}  
TreeNode\* levelOrderCreateTree(vector<string>& arr) { // used in make\_trees  
 if (arr.size() == 0)  
 return nullptr;  
 TreeNode \*head = new TreeNode(stoi(arr[0]));  
 createTree(head, 0, arr);  
 return head;  
}  
  
void make\_trees(vector<Product>prods, int am\_prods)  
{  
 for(int i =0; i< am\_prods;i++) {  
 cout << endl;  
 if(prods[i].name!=prods[i+1].name) // if one object in a tree  
 one\_num\_tree(prods[i]);  
 else {  
 vector<string>tmp;  
 while(prods[i].name==prods[i+1].name) // find objects with the same name to make one tree  
 {  
 tmp.push\_back(to\_string(prods[i].price));  
 i++;  
 }  
 tmp.push\_back(to\_string(prods[i].price));  
 if(prods[i].name!=prods[i+1].name) // if the next object doesn't belong to the tree  
 {  
 TreeNode\* tree = levelOrderCreateTree(tmp); // create the tree and write array in  
 cout<< prods[i].name<<":"<< endl;  
 printLevelOrder(tree); // output the tree  
  
 }  
  
 }  
 cout<< endl;  
 }  
}  
  
int check\_same(vector<Product>prods, Product good) // check if object is already in a list  
{  
 int i =0;  
 for (auto it = prods.begin();it!=prods.end();it++) {  
 if(it->name ==good.name && it->price == good.price)  
 return i;  
 i++;  
 }  
 return -1;  
}  
  
int find\_total(vector<Product>prods, string good) // count total for the product  
{  
 int total\_cost =0;  
 for (auto it = prods.begin();it!=prods.end();it++) {  
 if(it->name ==good)  
 total\_cost+=it->total;  
 }  
 return total\_cost;  
}

**Результат:**

****

Для об’єкту “Banana”(приклад):

****

****

**Висновок**: У цій лабораторній роботі було вивчено особливості організації і обробки дерев. Була постановлена задача, в якій визначався клас продуктів, введений користувачем та 1 структура дерева, створена програмою для організації інформації про ціни продуктів. У алгоритмі були уточнені обмеження з використанням умов при яких задача буде виконана правильно. За допомоги цієї лабораторної роботи можливо складання та сортування списку продуктів, рахування загальної вартості певного продукту, заданого користувачем, створення дерева для цін певних продуктів та їх вивід по рівням, додавання до цих дерев та списків нових об’єктів в необмеженій кількості.