**Міністерство освіти та науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра** **ІПІ (ІСТ)**

**Звіт**

з лабораторної роботи № 6 з дисципліни «Алгоритми та структури даних 2. Структури даних»

#### **«Деревовидні структури даних»**

**Виконав:**

*Студент I курсу*

*гр. ІП-з31*

Ткаченко К.О.

**Перевірила:**

к.т.н., Халус Олена Андріївна

2024

Зміст

1. **МЕТА ЛАБОРАТОРНОЇ РОБОТИ ..................................................... 3**
2. **ЗАВДАННЯ ............................................................................................... 4**
3. **ВИКОНАННЯ .......................................................................................... 4**

3.1 ПСЕВДОКОД АЛГОРИТМІВ ................................................. 8

3.2 ПРОГРАМНА РЕАЛІЗАЦІЯ ...................................................... 8

3.2.1 Вихідний код ........................................................................... 8

3.2.2 Приклади роботи ..................................................................... 8

**ВИСНОВОК** .......................................................................................................... 9

**КРИТЕРІЇ ОЦІНЮВАННЯ** ............................................................................. 10

1. МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні підходи формалізації та імплементації алгоритмів побудови та обробки базових деревовидних структур даних.

1. ЗАВДАННЯ

Розробити алгоритм розв’язання задачі відповідно до варіанту. Виконати програмну реалізацію задачі. Не використовувати вбудовані деревовидні структури даних (контейнери). Зробити висновок по лабораторній роботі.

23. Побудувати бінарне дерево для зберігання даних виду: найменування товару, його кількість, вартість одиниці. Забезпечити виконання операцій додавання нового елемента у дерево в діалоговому режимі та підрахунку загальної вартості вказаного товару.

1. ВИКОНАННЯ
   1. Псевдокод алгоритмів

Структура Product:

рядок name

ціле число quantity

дійсне число price

вказівник left на Product

вказівник right на Product

Конструктор Product(name, quantity, price):

Присвоїти значення name, quantity та price

Ініціалізувати left та right як nullptr

Деструктор ~Product():

Видалити left

Видалити right

Функція insertProduct(корінь, name, quantity, price):

current = адреса кореня

поки \*current не nullptr:

якщо name < (\*current)->name:

current = адреса (\*current)->left

інакше якщо name > (\*current)->name:

current = адреса (\*current)->right

інакше:

(\*current)->quantity += quantity

повернути корінь

\*current = новий Product(name, quantity, price)

повернути корінь

Функція calculateTotalCost(корінь, name):

поки корінь не nullptr:

якщо name == корінь->name:

повернути корінь->quantity \* корінь->price

якщо name < корінь->name:

корінь = корінь->left

інакше:

корінь = корінь->right

повернути 0

Функція printProducts(корінь):

створити стек s

current = корінь

поки current не nullptr або s не порожній:

поки current не nullptr:

додати current до s

current = current->left

current = верхній елемент s

вивести current->name, current->quantity та current->price

видалити верхній елемент з s

current = current->right

Функція main():

корінь = nullptr

повторювати:

вивести меню

отримати вибір від користувача

перемикач (вибір):

випадок 1:

отримати name, quantity та price від користувача

корінь = insertProduct(корінь, name, quantity, price)

випадок 2:

отримати searchName від користувача

totalCost = calculateTotalCost(корінь, searchName)

вивести totalCost

випадок 3:

printProducts(корінь)

випадок 4:

вивести "Exiting..."

в іншому випадку:

вивести "Invalid choice..."

до тих пір, поки вибір != 4

повернути 0

3.2 Програмна реалізація

3.2.1 Вихідний код

#include <iostream>

#include <limits>

#include <stack>

#include <string>

#include <utility>

using namespace std;

struct Product {

string name;

int quantity;

double price;

Product \*left;

Product \*right;

Product(string n, int q, double p) : name(std::move(n)), quantity(q), price(p), left(nullptr), right(nullptr) {

}

~Product() {

delete left;

delete right;

}

};

Product \*insertProduct(Product \*root, const string &name, int quantity, double price) {

Product \*\*current = &root;

while (\*current != nullptr) {

if (name < (\*current)->name) {

current = &((\*current)->left);

} else if (name > (\*current)->name) {

current = &((\*current)->right);

} else {

(\*current)->quantity += quantity;

return root;

}

}

\*current = new Product(name, quantity, price);

return root;

}

double calculateTotalCost(Product \*root, string name) {

while (root != nullptr) {

if (name == root->name) {

return root->quantity \* root->price;

}

if (name < root->name) {

root = root->left;

} else {

root = root->right;

}

}

return 0; // Return 0 if the product with the given name is not found

}

void printProducts(const Product \*root) {

stack<const Product \*> s;

const Product \*current = root;

while (current != nullptr || !s.empty()) {

while (current != nullptr) {

s.push(current);

current = current->left;

}

current = s.top();

s.pop();

cout << "Product: " << current->name << ", Quantity: " << current->quantity << ", Price: " << current->price <<

endl;

current = current->right;

}

}

int main() {

Product \*root = nullptr;

char choice;

string name;

int quantity;

double price;

string searchName;

double totalCost;

bool isValidInput = false;

do {

cout << "1 - Add product\n2 - Calculate the total cost of the product\n3 - List all products\n4 - Exit\n";

cin >> choice;

switch (choice) {

case '1':

cout << "Enter product name: ";

cin >> name;

try {

cout << "Enter quantity: ";

cin >> quantity;

if (cin.fail()) {

throw invalid\_argument("Invalid input. Please enter a valid integer for quantity.");

}

cout << "Enter price: ";

cin >> price;

if (cin.fail()) {

throw invalid\_argument("Invalid input. Please enter a valid double for price.");

}

root = insertProduct(root, name, quantity, price);

} catch (const invalid\_argument &e) {

cout << e.what() << endl;

cin.clear();

cin.ignore(numeric\_limits<int>::max(), '\n');

}

break;

case '2':

cout << "Enter the name of the product to calculate the total cost:";

cin >> searchName;

totalCost = calculateTotalCost(root, searchName);

cout << "Total cost of " << searchName << ": " << totalCost << endl;

break;

case '3':

cout << "Listing all products:\n";

printProducts(root);

break;

case '4':

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice. Please enter 1, 2, 3, or 4.\n";

break;

}

} while (choice != '4');

return 0;

}

3.2.2 Приклади роботи

На рисунку 3.1 показан приклад робоит програми.

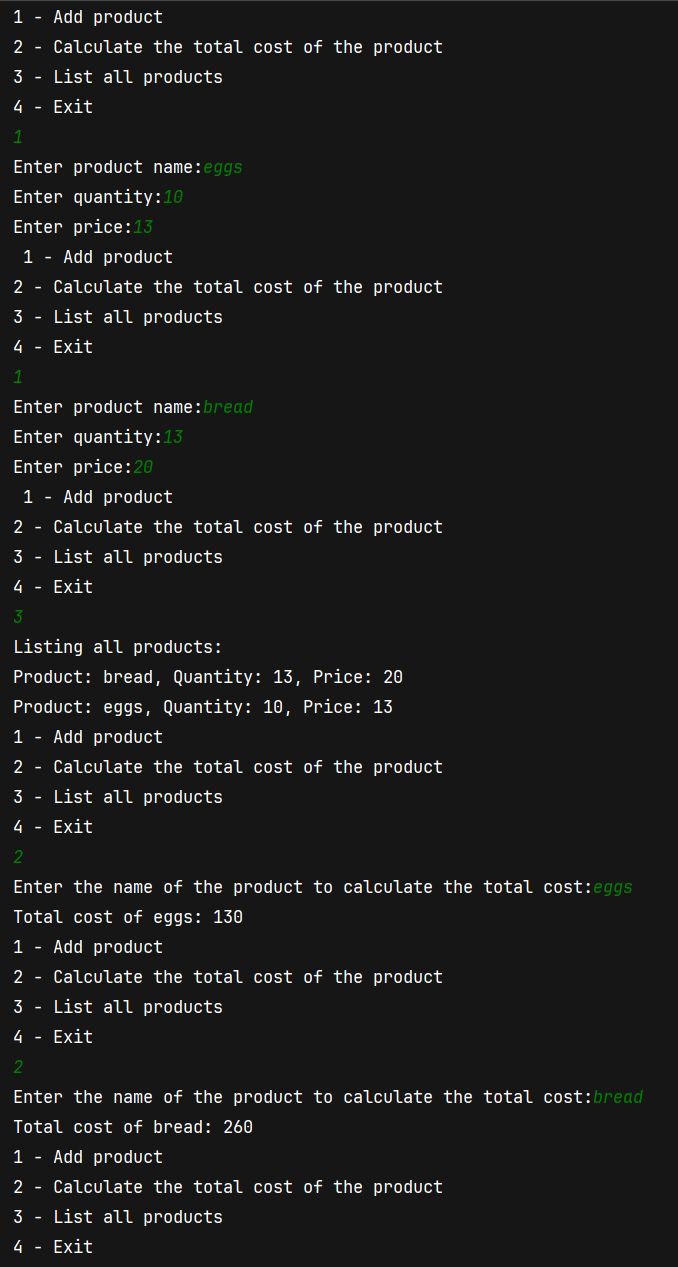


Рисунок 3.1

ВИСНОВОК

У даній роботі ми розглянули реалізацію простої системи управління продуктами за допомогою бінарного дерева. Програма дозволяє користувачу додавати продукти, обчислювати загальну вартість конкретного продукту та виводити список всіх продуктів.

У реалізації використано структуру даних бінарного дерева для зберігання продуктів. Кожен продукт має назву, кількість та ціну, а також посилання на лівого та правого наступників, що дозволяє побудувати бінарне дерево для швидкого доступу та маніпулювання даними.

Основні функції програми включають:

1. Додавання нового продукту до системи.
2. Обчислення загальної вартості конкретного продукту за заданою кількістю.
3. Виведення списку всіх продуктів у впорядкованому порядку.

В процесі роботи було використано концепції динамічного виділення пам'яті та обробки винятків для забезпечення коректної роботи програми та запобігання можливих помилок. Крім того, було використано стек для ітеративного обходу бінарного дерева та ефективного виводу всіх продуктів у відсортованому порядку.

Загалом, ця робота демонструє важливість використання структур даних та алгоритмів для розв'язання практичних задач у програмуванні, а також вміння розробляти ефективні та надійні програми з використанням сучасних методів розробки.