Учреждение образования Федерации профсоюзов Беларуси

«Международный университет «МИТСО»

Рег. № \_\_\_\_\_\_\_\_\_\_ Кафедра информационных

Дата \_\_\_\_\_\_\_\_ 2024 г. технологий

**КУРСОВОЙ ПРОЕКТ**

на тему:

**«** **»**

по дисциплине: Объектно-ориентированное проектирование и программирование

|  |  |
| --- | --- |
| Основные замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_    Отметка о допуске курсового проекта к защите:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.  Подпись научного руководителя: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_                                         (подпись)  Борисов Александр Николаевич  Курс 2 группа 2220  Факультет: экономический;  Специальность: информационные системы и технологии;  Научный руководитель:  старший преподаватель  Жалов Александр Петрович  апфС |

Минск, 2024

# Учреждение образования Федерации профсоюз Беларуси

**«Международный университет «МИТСО» Экономический факультет**

# Кафедра информационных технологий

**Специальность 6-05-0611-01 Информационные системы и технологии**

УТВЕРЖДАЮ

И.о. заведующего кафедрой информационных технологий

А.П.Жалов

« » 2024 г.

# ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

**Студент:** вставить свои Ф.И.О.

**Номер группы:** Указать номер группы

**Тема:** Указать свою тему

**Руководитель:** Жалов А.П. ст. преподаватель кафедры информационных технологий учреждения образования Федерации профсоюз Беларуси

«Международный университет «МИТСО» Дата выдачи задания: 12.02.2024 г.

Срок сдачи законченной работы: 29.04.2024 г.

Цель и содержание курсовой работы – закрепление теоретического материала и выработка навыков разработки алгоритма решения задачи и ее программной реализации.

Курсовая работа состоит из пояснительной записки и приложения(программы), реализующего задачу, поставленную в теме работы.

# Примерное содержание пояснительной записки:

Титульный лист;

Задание на курсовую работу; Реферат;

Содержание;

Введение (с указанием цели и основных задач для ее достижения). Рекомендуемая структура разделов и подразделов основной части: 1 Анализ предметной области

* 1. Анализ задачи
  2. Требования к разработке

1. Проектирование приложения
   1. Средства проектирования

Разработка схемы алгоритма программы 3 Разработка приложения

* 1. Структура программного проекта
  2. Разработка и описание приложения
  3. Тестирование приложения
  4. Инструкция пользователя Заключение.

Список использованных источников. Приложения.

В разделе «Анализ предметной области» приводится анализ задачи на основании выданного задания, описывается исследование предметной области. На основе изучения литературных источников отечественных и зарубежных авторов рассматривается сущность исследуемой проблемы, анализируются различные подходы их решения, дается их критический анализ, излагается собственная позиция исследователя.

Выявляются требования к разработке, а именно необходимо выбрать среду разработки, описать ее преимущества и недостатки, грамотно, аргументировано обосновать свой выбор. Разработать структуру будущей программы, выделить ее основные части и продумать правила взаимодействия.

Рекомендуемый объем этого раздела 7-10 страниц.

Раздел «Проектирование приложения» содержит описание процесса проектирования программного приложения на основе разработки алгоритма ее работы и графического его представления в виде схемы алгоритма программы.

Рекомендуемый объем этого раздела 5-7 страниц.

Раздел «Разработка приложения» содержит описание разработанного приложения, процесса разработки пользовательского интерфейса, инструкции пользователя и тестирование программного приложения.

Рекомендуемый объем этого раздела до 10 страниц.

Заключение должно содержать выводы по выполнению задания на курсовую работу и соответствовать введению в смысле достижения указанных в нем поставленной цели и задач проектирования. Следует отметить преимущества, связанные с реализацией проектных предложений, отметить недостатки работы, дать практические рекомендации по совершенствованию объекта проектирования, охарактеризовать перспективы дальнейшего развития работы. Рекомендуемый объем раздела 1-2 страницы.

Список использованных источников и количество приложений формально не ограничены. Рекомендуемый объем используемых литературных источников от 5 до 20 наименований. Сведения об использованных источниках должны быть выполнены в соответствии с требованиями ГОСТ 7.1-2003.

Приложение к курсовой работе должно включать листинг программного кода разработанного приложения.

Демонстрационная версия программного продукта предоставляется на носителе информации, который прилагается к пояснительной записке. Длительность демонстрации программного продукта при защите курсовой

работы составляет 5- 7 минут.

Для защиты курсовой работы рекомендуется использовать компьютерную презентацию в виде последовательности слайдов. Демонстрация графического и иллюстративного материала в этом случае выполняется на мультимедийном проекторе.

Руководитель

Задание принял к исполнению

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ 6**](#_Toc145063505)

[**1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7**](#_Toc145063505)

[**2. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ… 8**](#_Toc145063513)

[**3. ОПИСАНИЕ АЛГОРИТМА 10**](#_Toc145063521)

[**4. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 19**](#_Toc145063524)

[**5. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ 22**](#_Toc145063524)

[**ЗАКЛЮЧЕНИЕ 23**](#_Toc145063525)

[**СПИСОК ЛИТЕРАТУРЫ 24**](#_Toc145063525)

[**ПРИЛОЖЕНИЕ А 25**](#_Toc145063525)

**ВВЕДЕНИЕ**

Проект “Программное приложение для управления продуктовыми данными” нацелен на создание мощного инструмента, который упростит обработку и анализ больших объемов продуктовой информации.

Это приложение станет незаменимым помощником для компаний, работающих в сфере ритейла и маркетингового анализа. Вот ключевые задачи, которые были определены для достижения поставленной цели:

• Разработка пользовательского интерфейса: Первоочередная задача - создать интуитивно понятный интерфейс, который обеспечит простое и эффективное взаимодействие с системой. Интерфейс должен предлагать функции для визуализации продуктовых данных, удобного поиска и доступа к разнообразным функциям управления продуктами.

• Программирование логики обработки данных: Сердцем приложения является его способность к быстрой и точной обработке продуктовых данных. Необходимо использовать свои знания в области алгоритмов и структур данных для создания эффективного приложения, которое сможет выполнять различные операции - от фильтрации и категоризации до выполнения сложных запросов и аналитических расчетов. Важно также обеспечить возможность интеграции с другими системами и базами данных для обмена информацией.

• Тестирование и отладка: Очень важным этапом является тестирование всех аспектов приложения для обеспечения его стабильности и удобства использования. Тестирование должно охватывать проверку функциональности, производительности, а также удобства пользовательского интерфейса. Особое внимание следует уделить защите данных и предотвращению неавторизованного доступа.

Приложение для управления продуктовыми данными должно быть не только функционально полным, но и максимально простым и понятным для пользователя, что делает процесс его разработки особенно сложным и захватывающим. Успешная реализация этого проекта предоставит пользователям уникальный инструмент для эффективного управления продуктовыми данными, способствующий оптимизации бизнес-процессов и повышению уровня обслуживания клиентов.

1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

При разработке программного приложения для управления продуктовыми данными, важно учесть ряд ключевых аспектов и требований, чтобы обеспечить его высокую эффективность и удобство в использовании. Основные функциональные требования к такому приложению должны включать механизмы для добавления и управления информацией о продуктах. Это включает в себя возможность ввода данных о продуктах, связанных с их характеристиками и ценами, а также управление этой информацией, включая ее обновление или удаление в случае изменения ассортимента. Важно также предусмотреть функции поиска, чтобы пользователи могли быстро находить необходимые продукты среди большого количества данных.

Анализ данных является критически важной функцией, позволяющей проводить статистический анализ и выявлять тенденции в потреблении продуктов, что может быть полезно для оптимизации запасов и планирования продаж. Дополнительные функции могут включать визуализацию данных, представляющую информацию о продуктах в виде графиков, диаграмм и других графических элементов. Управление доступом к данным имеет решающее значение для защиты конфиденциальной информации и предотвращения несанкционированного доступа. Интеграция с внешними базами данных и сервисами может обогатить информацию о продуктах и повысить ее актуальность. Существующие решения в этой области могут варьироваться от базовых инструментов управления данными до сложных систем, разработанных для нужд розничных сетей или компаний, занимающихся распределением товаров.

При создании приложения для управления продуктовыми данными также крайне важно обратить внимание на аспекты безопасности, включая защиту информации и механизмы контроля доступа, чтобы гарантировать конфиденциальность и защиту от неправомерного использования. В конечном итоге, идеальное приложение для управления продуктовыми данными должно сочетать в себе обширный набор функций с высоким уровнем безопасности и простотой использования, позволяя пользователям эффективно управлять информацией о продуктах и проводить ее анализ в соответствии с бизнес-потребностями.

1. **ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ**

Для создания моего программного приложения я выбрал язык программирования C++, который широко известен своей мощью, гибкостью и эффективностью. C++ является объектно-ориентированным языком программирования, что позволяет создавать модульные и легко расширяемые программы. Его высокая производительность и возможности непосредственного управления памятью делают его идеальным выбором для разработки приложений, требующих высокой производительности и эффективного использования ресурсов компьютера.

Одним из ключевых преимуществ C++ является его высокая скорость выполнения. Это особенно важно для приложений, работающих с большими объемами данных, такими как приложения для обработки данных об почтовых индексов. Благодаря оптимизации кода и возможности использования низкоуровневых инструкций процессора, приложения на C++ могут обрабатывать данные быстрее и эффективнее, что повышает производительность и ускоряет процесс работы.

Кроме того, C++ обладает обширными библиотеками и фреймворками, которые облегчают разработку приложений. Например, стандартная библиотека шаблонов (STL) предоставляет широкий набор готовых компонентов, таких как контейнеры, алгоритмы и итераторы, которые значительно упрощают написание кода и ускоряют разработку.

Благодаря этим библиотекам и фреймворкам разработчики могут сосредоточиться на решении конкретных задач, минимизируя время, затраченное на разработку и отладку.

В процессе разработки я использовал среду разработки Visual Studio, которая предоставляет широкий набор инструментов и функций для создания приложений на C++.

Итак, выбор C++ для разработки моего приложения оказался удачным, благодаря его мощности, эффективности и обширным возможностям для создания высокопроизводительных и легко масштабируемых приложений. Среда разработки Visual Studio дополнила этот выбор, предоставляя удобные инструменты для создания пользовательского интерфейса и управления процессом разработки.

Для того, чтобы начать проектировать приложение, еще недостаточно выбрать язык программирования, среду разработки и технологию.

1. **ОПИСАНИЕ АЛГОРИТМА**

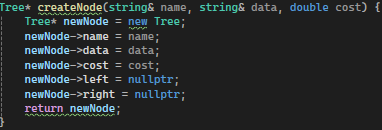
Для создания и программирования функционала приложения, я создал следующие функции и параметры.

Параметры:

* struct Tree

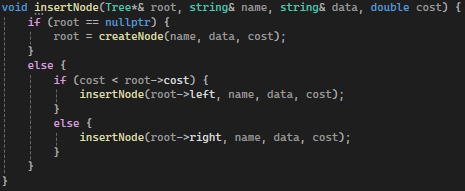
Функции:

* createNode - создает новый узел дерева.



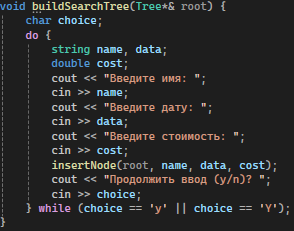
*Рис 3.1 – Код функции createNode*

* insertNode - для добавления узла в двоичное дерево поиска



*Рис 3.2 – Код функции insertNode*

* buildSearchTree - определяет функцию для создания двоичного дерева поиска, запрашивая данные у пользователя.



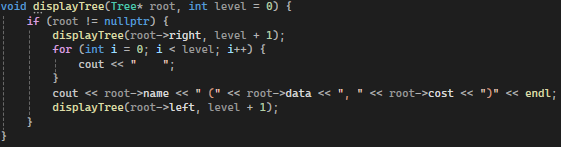
*Рис 3.3 – Код функции buildSearchTree*

* createEmptyTree - определяет функцию для инициализации пустого двоичного дерева поиска.



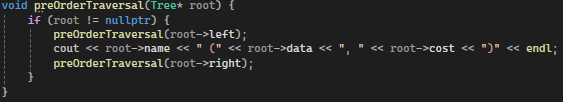
*Рис 3.4 – Код функции createEmptyTree*

* displayTree - функция предназначена для отображения элементов двоичного дерева поиска в виде дерева.



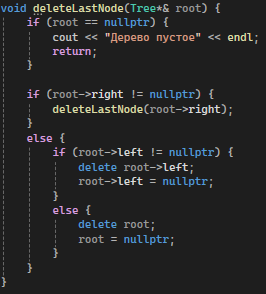
*Рис 3.5 – Код функции displayTree*

* preOrderTraversal - предназначена для обхода и печати элементов двоичного дерева поиска в порядке предварительного обхода.



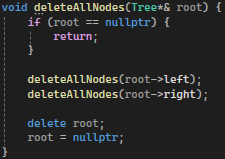
*Рис 3.6 – Код функции preOrderTraversal*

* deleteLastNode - удаление последнего элемента, добавленного в дерево.



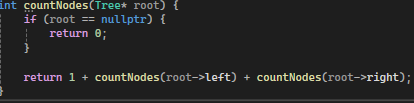
*Рис 3.7 – Код функции deleteLastNode*

* deleteAllNodes - предназначена для рекурсивного удаления всех узлов в двоичном дереве поиска.



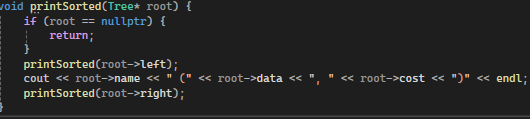
*Рис 3.8 – Код функции deleteAllNodes*

* countNodes - предназначена для подсчета количества узлов в двоичном дереве поиска.



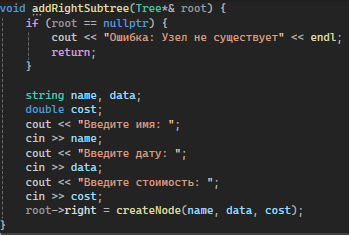
*Рис 3.9 – Код функции countNodes*

* printSorted - функция предназначена для отображения элементов двоичного дерева поиска в отсортированном виде.



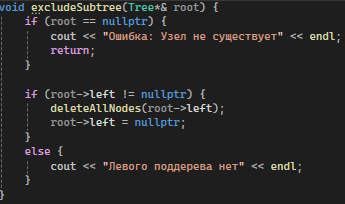
*Рис 3.10 – Код функции printSorted*

* addRightSubtree - предназначена для добавления правого поддерева к существующему узлу в двоичном дереве поиска.



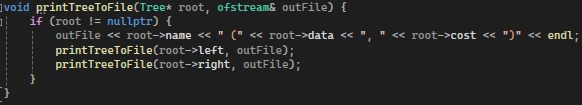
*Рис 3.11 – Код функции addRightSubtree*

* excludeSubtree - предназначена для удаления левого поддерева существующего узла в двоичном дереве поиска.



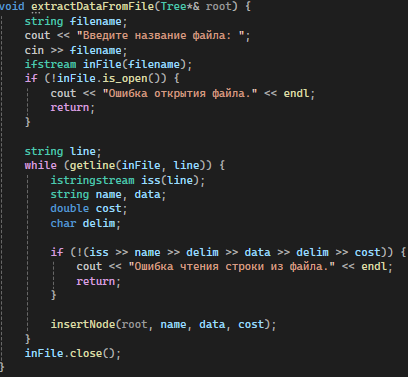
*Рис 3.12 – Код функции excludeSubtree*

* printTreeToFile - предназначена для печати двоичного дерева поиска (BST) в текстовый файл. Она использует рекурсивный подход для обхода дерева и записи информации о каждом узле в файл.



*Рис 3.13 – Код функции printTreeToFile*

* extractDataFromFile - предназначена для извлечения данных из текстового файла и создания двоичного дерева поиска на основе этих данных.



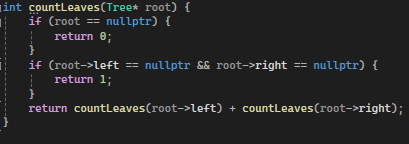
*Рис 3.14 – Код функции extractDataFromFile*

* findMaxCostProducts - функция предназначена для поиска и вывода всех узлов в двоичном дереве поиска, которые имеют максимальную стоимость.

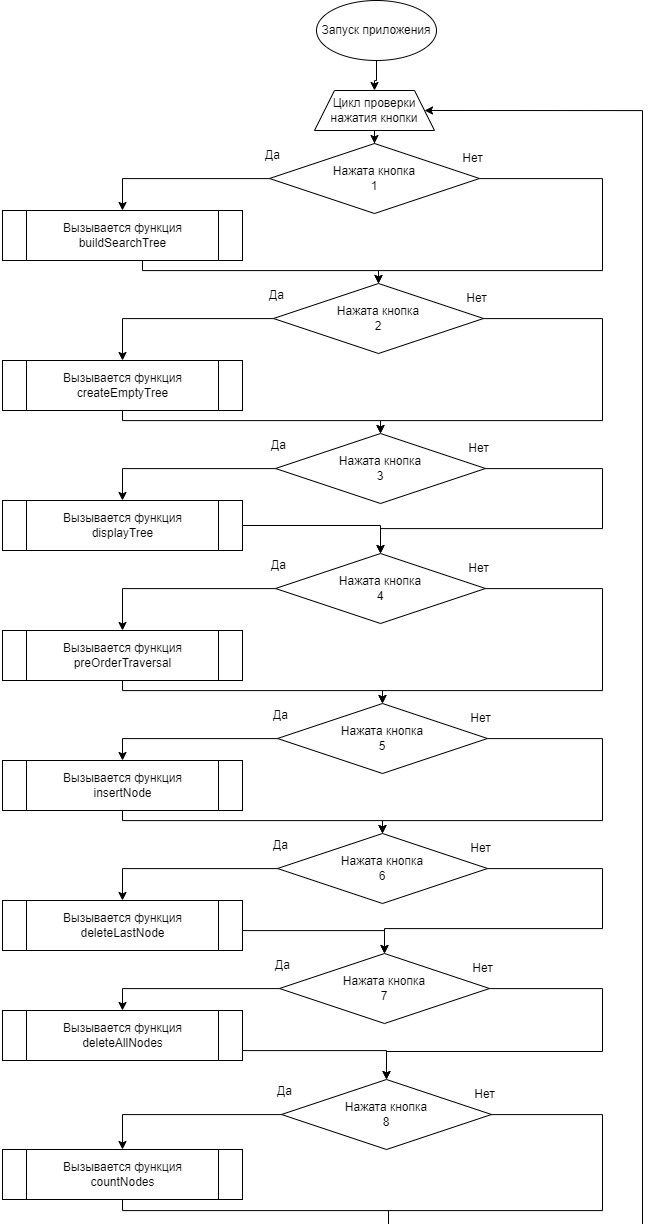


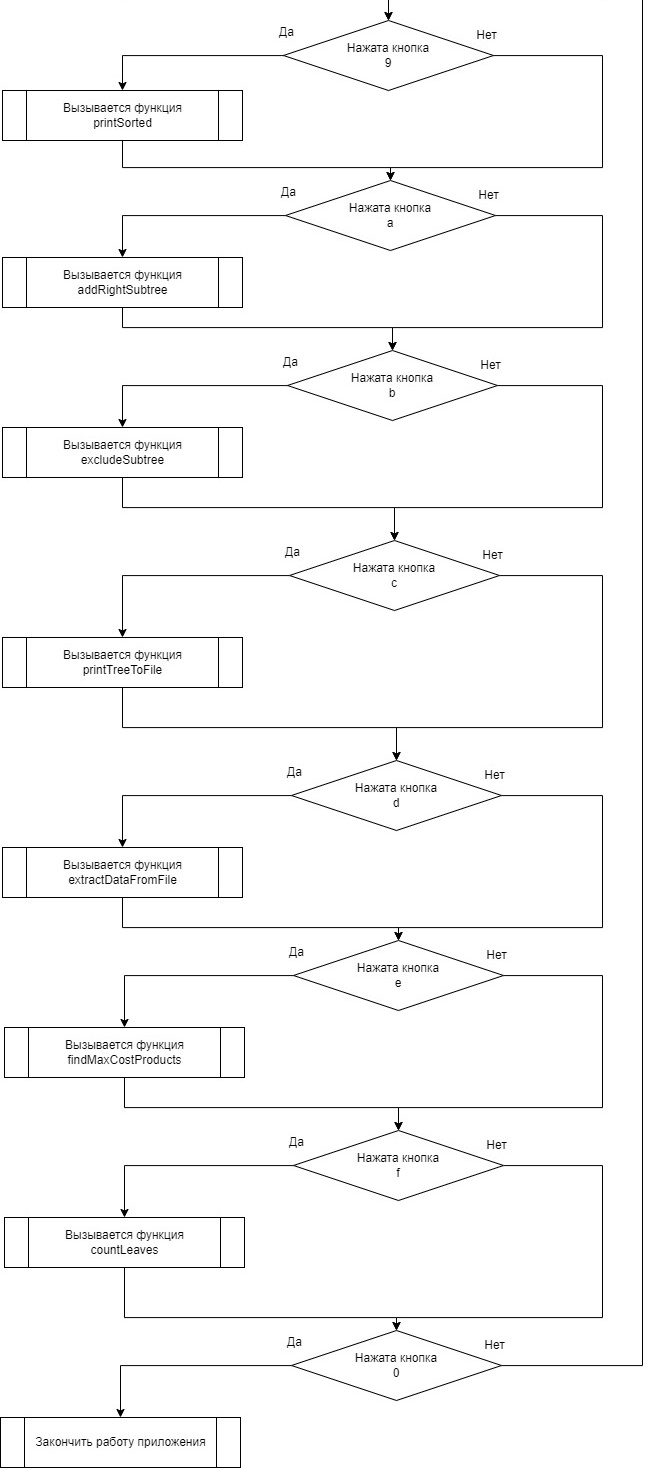
*Рис 3.15 – Код функции findMaxCostProducts*

* countLeaves - функция предназначена для подсчета количества листовых узлов в двоичном дереве поиска.



*Рис 3.16 – Код функции countLeaves*

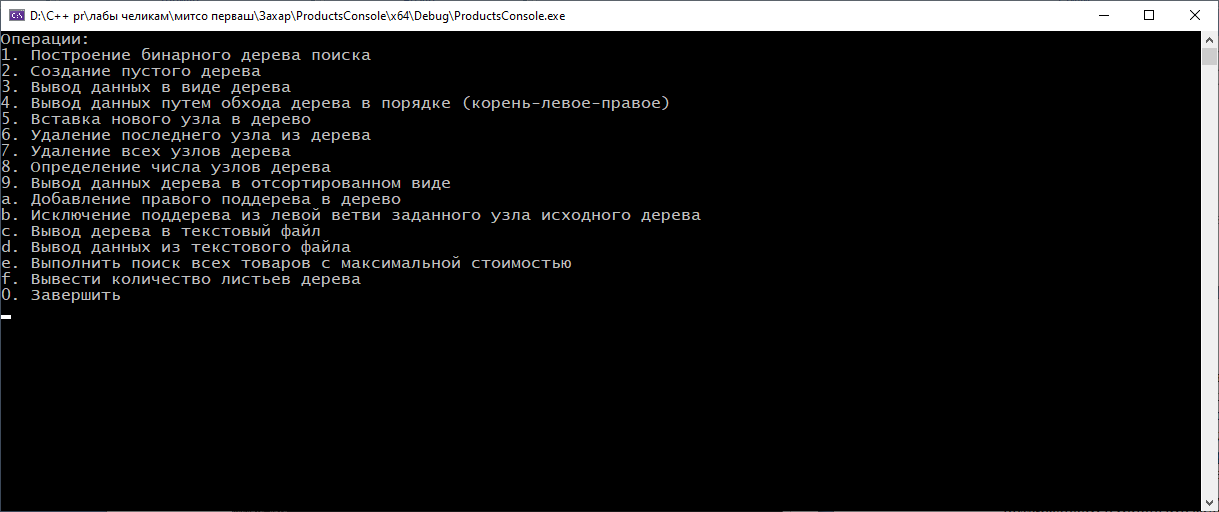




*Рис 3.15 – Диаграмма работы приложения*

1. **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

При запуске приложения, вам откроется консоль и выводится меню с выбором определенных операций.



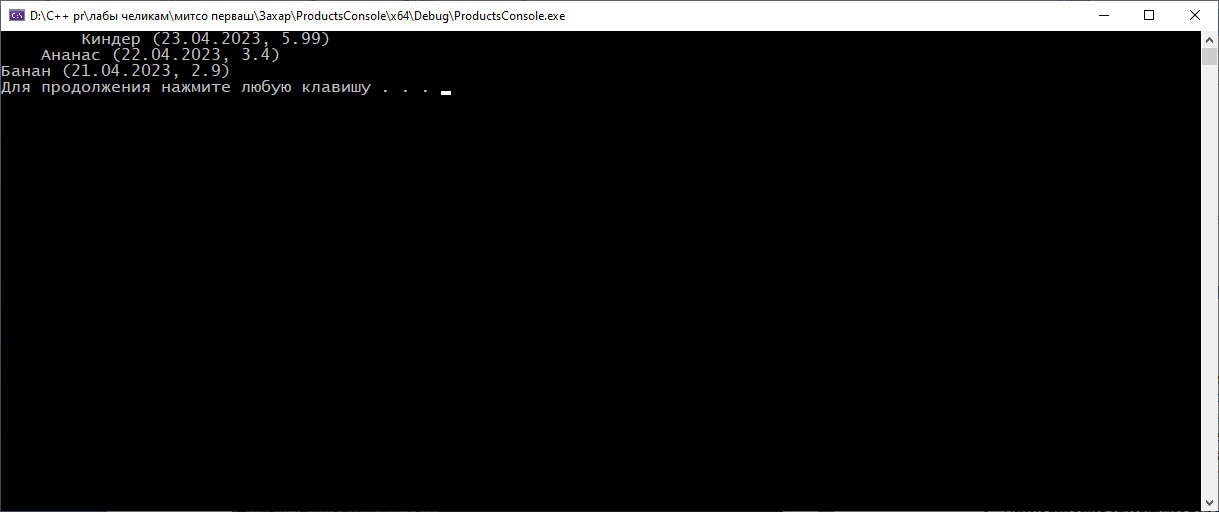
*Рис 4.1 – Меню приложения*

При выборе первого пункта мы можем сколько угодно раз добовлять информацию о продуктах в наше дерево.



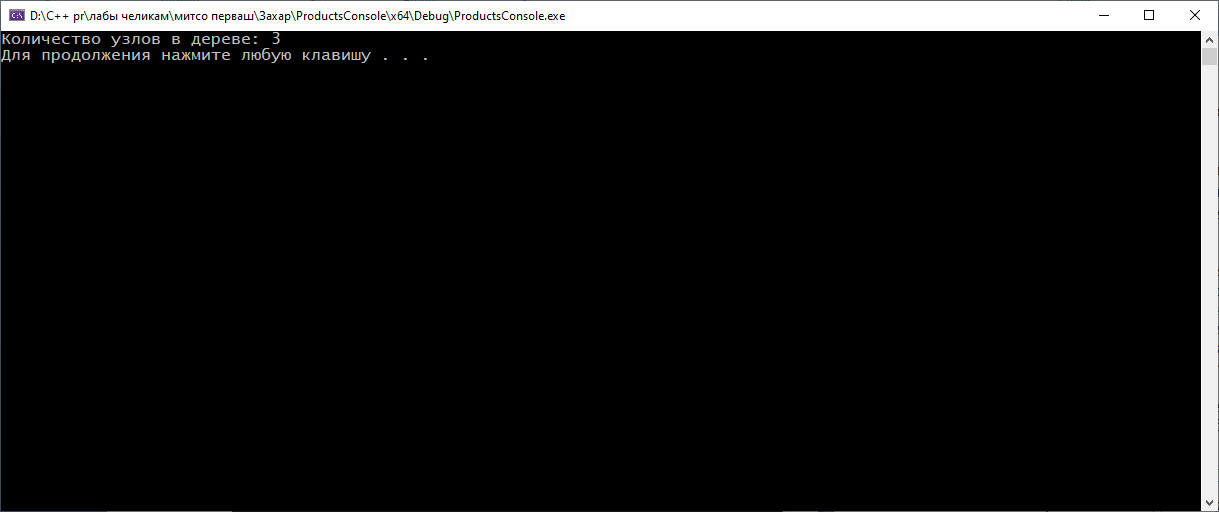
*Рис 4.2 – Добавление информации в бинарное дерево*

При выборе 3го пункта, нам выведиться информация в виде дерева.



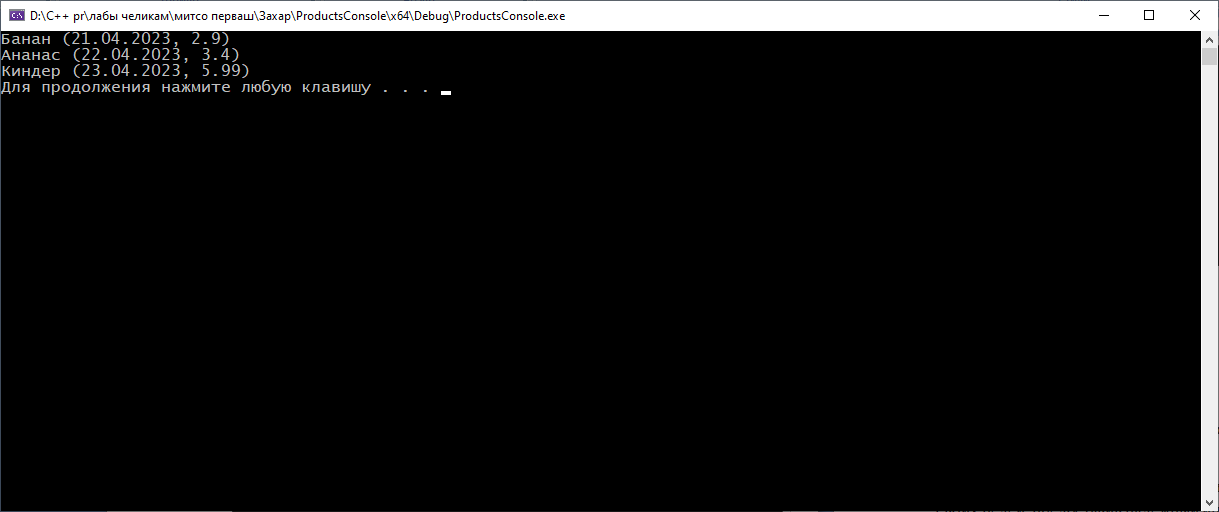
*Рис 4.3 – Вывод информации из бинарного дерева в консоль в виде дерева*

При выборе пунтка f, нам выведиться количество листев дерева.



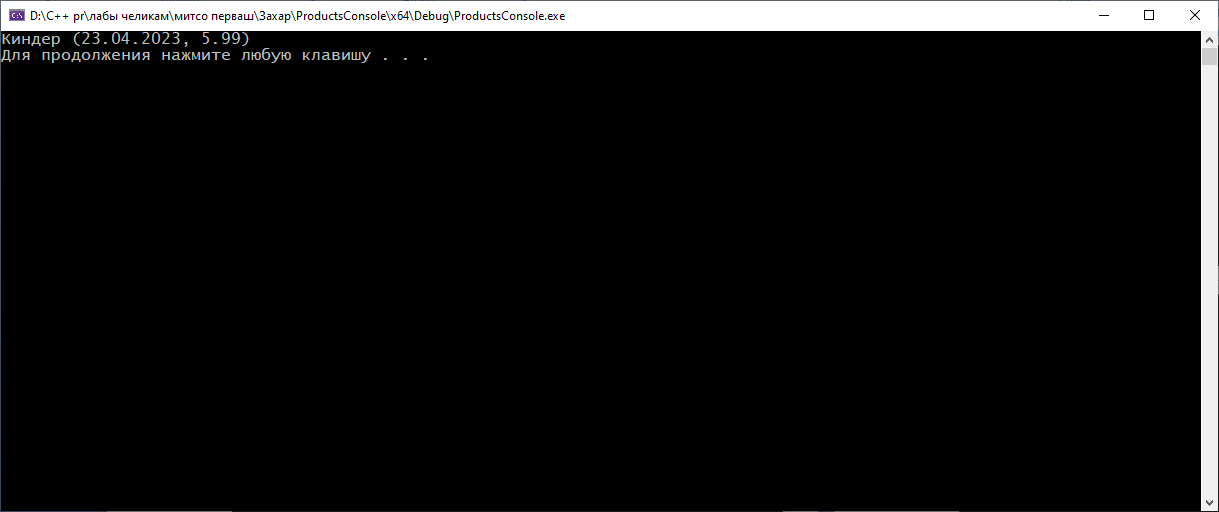
*Рис 4.4 – Вывод информации про количество листьев дерева*

При выборе пункта 4, вся информация дерева выведиться в столбик.



*Рис 4.5 – Вывод информации дерева в столбик*

При выборе пункта е, вам выведиться список товаров с максимальной стоимостью.



*Рис 4.6 – Вывод информации дерева в столбик*

1. **РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ**

Процесс тестирования функционала приложения был тщательно спланирован и организован для обеспечения его надежности и эффективности. В ходе подготовки были созданы тестовые данные, охватывающие различные сценарии использования, включая стандартные операции, граничные условия и исключительные ситуации. Это позволило полностью покрыть все возможные варианты работы приложения и подтвердить корректность всех предусмотренных операций, таких как создание дерева, вставка и удаление узлов, а также обход дерева.

Каждая функция приложения была тщательно протестирована в отдельности с использованием подготовленных тестовых данных. Результаты тестирования показали, что функции полностью соответствуют ожиданиям и корректно обрабатывают неправильные входные данные, а также правильно реагируют на исключительные ситуации.

Интеграционное тестирование показало, что взаимодействие различных функций приложения друг с другом осуществляется без ошибок, что подтверждает высокое качество кода. Также было проведено тестирование граничных условий, в ходе которого приложение успешно справилось с экстремальными значениями параметров, не выявив проблем с выходом за границы массивов и некорректными указателями.

Тестирование производительности выявило, что время выполнения функций при работе с различными объемами данных соответствует высоким стандартам производительности, что свидетельствует об эффективном использовании ресурсов компьютера и хорошей оптимизации программы.

Негативные тесты, направленные на выявление ошибок, также были успешно проведены. Передача некорректных значений аргументов функций не привела к сбоям, что говорит о высокой устойчивости приложения к ошибкам.

В заключение, тестирование пользовательского интерфейса подтвердило его удобство использования, соответствие дизайну и корректность отображения информации. Интуитивно понятный и удобный интерфейс обеспечивает положительный пользовательский опыт, что является важным фактором для успеха любого программного продукта.

**ЗАКЛЮЧЕНИЕ**

В ходе разработки приложения для обработки данных были успешно достигнуты поставленные цели, описанные в начале проекта.

В части разработки пользовательского интерфейса было создано интуитивно понятное и функциональное графическое окружение. Главное окно приложения адаптировано под различные категории пользователей, обеспечивая удобство и эффективность работы. Использование среды разработки Visual Studio и ее компонента, позволило легко и быстро создавать различные функции, обеспечивая простоту взаимодействия с приложением.

В области программирования логики обработки данных была разработана эффективная система обработки информации. Благодаря глубоким знаниям в области алгоритмов и структур данных, приложение обеспечивает быструю и точную обработку данных об почтовых индексов населения, а также предусматривает различные функции обработки, такие как импорт, фильтрация и визуализация данных.

Были успешно выполнены все поставленные цели для разработки приложения для обработки данных об почтовых индексов на языке программирования C++ с использованием среды разработки Visual Studio и приложение готово к использованию.

**СПИСОК ЛИТЕРАТУРЫ**

1. Создание блок-схем - <https://online.visual-paradigm.com/ru/w/ecbembgc/login.jsp?r=drive/#infoart:proj=0&dashboard>
2. Бертран Мейер «Почувствуй класс. Учимся программировать хорошо с объектами и контрактами» - <https://vk.com/doc9385624_274277026?hash=a351256a10ddc59413&dl=c31b6571a7272d7c7e>
3. Сайт по изучению С++ - [https://www.sololearn.com](https://www.sololearn.com/en/pagenotfound)
4. Скотт Мейерс. Эффективное использование C++
5. Язык программирования C++. Краткий курс - Бьерн Страуструп

**ПРИЛОЖЕНИЕ А**

main.cpp

#include <iostream>

#include <string>

#include <conio.h>

#include <Windows.h>

#include <fstream>

#include <sstream>

#include <stack>

using namespace std;

struct Tree {

string name;

string data;

double cost;

Tree\* left, \* right;

};

Tree\* createNode(string& name, string& data, double cost) {

Tree\* newNode = new Tree;

newNode->name = name;

newNode->data = data;

newNode->cost = cost;

newNode->left = nullptr;

newNode->right = nullptr;

return newNode;

}

void insertNode(Tree\*& root, string& name, string& data, double cost) {

if (root == nullptr) {

root = createNode(name, data, cost);

}

else {

if (cost < root->cost) {

insertNode(root->left, name, data, cost);

}

else {

insertNode(root->right, name, data, cost);

}

}

}

void buildSearchTree(Tree\*& root) {

char choice;

do {

string name, data;

double cost;

cout << "Введите имя: ";

cin >> name;

cout << "Введите дату: ";

cin >> data;

cout << "Введите стоимость: ";

cin >> cost;

insertNode(root, name, data, cost);

cout << "Продолжить ввод (y/n)? ";

cin >> choice;

} while (choice == 'y' || choice == 'Y');

}

void createEmptyTree(Tree\*& root) {

root = nullptr;

}

void displayTree(Tree\* root, int level = 0) {

if (root != nullptr) {

displayTree(root->right, level + 1);

for (int i = 0; i < level; i++) {

cout << " ";

}

cout << root->name << " (" << root->data << ", " << root->cost << ")" << endl;

displayTree(root->left, level + 1);

}

}

void preOrderTraversal(Tree\* root) {

if (root != nullptr) {

preOrderTraversal(root->left);

cout << root->name << " (" << root->data << ", " << root->cost << ")" << endl;

preOrderTraversal(root->right);

}

}

void deleteLastNode(Tree\*& root) {

if (root == nullptr) {

cout << "Дерево пустое" << endl;

return;

}

if (root->right != nullptr) {

deleteLastNode(root->right);

}

else {

if (root->left != nullptr) {

delete root->left;

root->left = nullptr;

}

else {

delete root;

root = nullptr;

}

}

}

void deleteAllNodes(Tree\*& root) {

if (root == nullptr) {

return;

}

deleteAllNodes(root->left);

deleteAllNodes(root->right);

delete root;

root = nullptr;

}

int countNodes(Tree\* root) {

if (root == nullptr) {

return 0;

}

return 1 + countNodes(root->left) + countNodes(root->right);

}

void printSorted(Tree\* root) {

if (root == nullptr) {

return;

}

printSorted(root->left);

cout << root->name << " (" << root->data << ", " << root->cost << ")" << endl;

printSorted(root->right);

}

void addRightSubtree(Tree\*& root) {

if (root == nullptr) {

cout << "Ошибка: Узел не существует" << endl;

return;

}

string name, data;

double cost;

cout << "Введите имя: ";

cin >> name;

cout << "Введите дату: ";

cin >> data;

cout << "Введите стоимость: ";

cin >> cost;

root->right = createNode(name, data, cost);

}

void excludeSubtree(Tree\*& root) {

if (root == nullptr) {

cout << "Ошибка: Узел не существует" << endl;

return;

}

if (root->left != nullptr) {

deleteAllNodes(root->left);

root->left = nullptr;

}

else {

cout << "Левого поддерева нет" << endl;

}

}

void printTreeToFile(Tree\* root, ofstream& outFile) {

if (root == nullptr) {

outFile << "Дерево пустое" << endl;

return;

}

stack<Tree\*> nodeStack;

nodeStack.push(root);

while (!nodeStack.empty()) {

Tree\* current = nodeStack.top();

nodeStack.pop();

outFile << current->name << " (" << current->data << ", " << current->cost << ")" << endl;

if (current->right != nullptr) {

nodeStack.push(current->right);

}

if (current->left != nullptr) {

nodeStack.push(current->left);

}

}

}

void extractDataFromFile(Tree\*& root) {

string filename;

cout << "Введите название файла: ";

cin >> filename;

filename += ".txt";

ifstream inFile(filename);

if (!inFile.is\_open()) {

cout << "Ошибка открытия файла." << endl;

return;

}

string line;

while (getline(inFile, line)) {

string name, data;

double cost;

char delim;

cout << line << endl;

}

inFile.close();

}

void findMaxCostProducts(Tree\* root) {

if (root == nullptr) {

cout << "Дерево пустое." << endl;

return;

}

double maxCost = root->cost;

stack<Tree\*> nodeStack;

nodeStack.push(root);

while (!nodeStack.empty()) {

Tree\* current = nodeStack.top();

nodeStack.pop();

if (current->cost > maxCost) {

maxCost = current->cost;

}

if (current->left != nullptr) {

nodeStack.push(current->left);

}

if (current->right != nullptr) {

nodeStack.push(current->right);

}

}

nodeStack.push(root);

while (!nodeStack.empty()) {

Tree\* current = nodeStack.top();

nodeStack.pop();

if (current->cost == maxCost) {

cout << current->name << " (" << current->data << ", " << current->cost << ")" << endl;

}

if (current->left != nullptr) {

nodeStack.push(current->left);

}

if (current->right != nullptr) {

nodeStack.push(current->right);

}

}

}

int countLeaves(Tree\* root) {

if (root == nullptr) {

return 0;

}

if (root->left == nullptr && root->right == nullptr) {

return 1;

}

return countLeaves(root->left) + countLeaves(root->right);

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Tree\* root = nullptr;

while (1) {

system("cls");

cout << "Операции:" << endl;

cout << "1. Построение бинарного дерева поиска" << endl;

cout << "2. Создание пустого дерева" << endl;

cout << "3. Вывод данных в виде дерева" << endl;

cout << "4. Вывод данных путем обхода дерева в порядке (корень-левое-правое)" << endl;

cout << "5. Вставка нового узла в дерево" << endl;

cout << "6. Удаление последнего узла из дерева" << endl;

cout << "7. Удаление всех узлов дерева" << endl;

cout << "8. Определение числа узлов дерева" << endl;

cout << "9. Вывод данных дерева в отсортированном виде" << endl;

cout << "a. Добавление правого поддерева в дерево" << endl;

cout << "b. Исключение поддерева из левой ветви заданного узла исходного дерева" << endl;

cout << "c. Вывод дерева в текстовый файл" << endl;

cout << "d. Вывод данных из текстового файла" << endl;

cout << "e. Выполнить поиск всех товаров с максимальной стоимостью" << endl;

cout << "f. Вывести количество листьев дерева" << endl;

cout << "0. Завершить" << endl;

switch (\_getch())

{

case '1':

system("cls");

buildSearchTree(root);

system("pause");

break;

case '2':

system("cls");

createEmptyTree(root);

system("pause");

break;

case '3':

system("cls");

displayTree(root);

system("pause");

break;

case '4':

system("cls");

preOrderTraversal(root);

system("pause");

break;

case '5':

system("cls");

{

string name, data;

double cost;

cout << "Введите имя: ";

cin >> name;

cout << "Введите дату: ";

cin >> data;

cout << "Введите стоимость: ";

cin >> cost;

insertNode(root, name, data, cost);

}

system("pause");

break;

case '6':

system("cls");

deleteLastNode(root);

system("pause");

break;

case '7':

system("cls");

deleteAllNodes(root);

system("pause");

break;

case '8':

system("cls");

cout << "Количество узлов в дереве: " << countNodes(root) << endl;

system("pause");

break;

case '9':

system("cls");

printSorted(root);

system("pause");

break;

case 'a':

system("cls");

addRightSubtree(root);

system("pause");

break;

case 'b':

system("cls");

excludeSubtree(root);

system("pause");

break;

case 'c':

system("cls");

{

string name;

cout << "Введите имя файла: ";

cin >> name;

name += ".txt";

ofstream file(name);

if (file.is\_open()) {

printTreeToFile(root, file);

file.close();

}

else {

cout << "Ошибка открытия файла." << endl;

}

}

system("pause");

break;

case 'd':

system("cls");

extractDataFromFile(root);

system("pause");

break;

case 'e':

system("cls");

findMaxCostProducts(root);

system("pause");

break;

case 'f':

system("cls");

cout << "Количество листьев в дереве: " << countLeaves(root) << endl;

system("pause");

break;

case '0':

return 0;

break;

default:

break;

}

}

}