Министерство образования и науки РФ  
Федеральное государственное автономное  
образовательное учреждение высшего профессионального образования  
«НИТУ МИСиС»

Институт ИТКН

**Курсовая работа**

По дисциплине«Технология программирования»

Тема: «Мониторинг торговых точек. Контроль за остатками товара в торговых точках. Мониторинг цен на товар в торговых точках.»

Выполнил:  
студент гр. БПИ-20-2  
Бобров М. А.

Проверил:   
преп. Широков А. И.

Москва, 2021 г.

**Содержание**

[Рабочий проект 3](#_Toc90331917)

[1 Разработка программы 3](#_Toc90331918)

[1.1 Программирование и отладка программы 3](#_Toc90331919)

[2 Испытания программы 22](#_Toc90331920)

[2.1 Корректировка программы и программной документации по результатам испытаний 22](#_Toc90331921)

[Описание программы 23](#_Toc90331922)

[Инструкция по эксплуатации программы 41](#_Toc90331923)

[1 Главное меню программы 41](#_Toc90331924)

[2 Меню вывода позиций 42](#_Toc90331925)

[2.1 Вывод всех позиций 43](#_Toc90331926)

[2.2 вывод определенной позиции 43](#_Toc90331927)

[3 Меню изменения позиции 44](#_Toc90331928)

[3.1 Изменение кода товара 45](#_Toc90331929)

[3.2 Изменение наименования товара 45](#_Toc90331930)

[3.3 Изменение количества товара 46](#_Toc90331931)

[3.4 Изменение цены товара 47](#_Toc90331932)

[4 Меню добавления позиции 47](#_Toc90331933)

Рабочий проект

1 Разработка программы

1.1 Программирование и отладка программы

Программирование данной программы осуществлялось на языке C++ в рабочей среде Visual Studio. Вся программа занимает 943 строчек кода в главном файле .cpp и 26 строчек кода в файле .h. Весь код подразделяется на главные и дополнительные модули, которые помогают в работе главным модулям путем оптимизации и улучшения читабельности кода. В главных модулях числятся: int main(), void ShowPositions(), void RedactPosition(), void AddPosition(), void DeletePosition(). В дополнительных модулях числятся: void ReadFile(), void WriteFile(), void CoutSpecificPosition(int Code), int GetMaxColLength(int Index), std::string FindPositionByCode(std::string Code), int ReturnChoiceInMain(std::string InterIndex), bool bIsDigit(std::string Input). Также присутствует переменная для чтения и записи файла std::fstream File и вектор для манипуляции с данными файла std::vector<std::vector<std::string>> TempFile{}. Ниже будет подробно рассказано про каждый модуль с листингом его кода.

Модуль int main() является самым главным модулем данной программы. Через него происходит запуск и завершение работы программы. В нем находится настройка на вывод в консоль русского языка, приветствие и выбор действия. Благодаря выбору действия программа переходит в другие модули.

Листинг кода int main():

int main()

{

//Для русского языка

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << endl << "Добро пожаловать в ShowKeeper!" << endl;

string InterIndex;

while (InterIndex != "exit")

{

//Подстраховка

File.close();

TempFile.clear();

InterIndex.clear();

cout << endl << "Введите индекс выбора действия или введите |exit| для выхода из прокраммы:" << endl;

cout << "1) ввежите 1 или |Show positions| чтобы перейти в окно вывода данных файла;" << endl;

cout << "2) введите 2 или |Redact positions| чтобы перейти в окно изменения позиции;" << endl;

cout << "3) введите 3 или |Add positions| чтобы перейти в окно добавления позиции;" << endl;

cout << "4) введите 4 или |Delete positions| чтобы перейти в окно удаления позиции." << endl;

cout << "Подсказка: ввод |exit| в любой функции кроме меню вернет вас обратно в меню." << endl;

getline(cin, InterIndex);

if (InterIndex == "exit")

{

return 0;

}

switch (ReturnChoiceInMain(InterIndex))

{

case 1:

{

ShowPositions();

break;

}

case 2:

{

RedactPosition();

break;

}

case 3:

{

AddPosition();

break;

}

case 4:

{

DeletePosition();

break;

}

default:

cout << endl << "ОШИБКА: неверный ввод! Введите индекс с 1 по 4 или указанные ключевые фразы!" << endl;

break;

}

};

}

Главный модуль ShowPositions() предназначен для вывода пользователю содержимого файла. Он использует дополнительный модуль ReadFile(), благодаря которому возможно чтение файла. Дополнительный модуль будет описан ниже. Далее идет выбор действия, где ожидается, что пользователь выберет вывод данных из всего файла или из определенной строки. При выводе всего файла читается весь вектор, который хранит данные в модуле. При выборе определенной строки считывается код товара, по коду товара в векторе ищется индекс данной строки. При существующей строке происходит вывод данных. Иначе происходит искусственная ошибка и пользователю предлагается повторить действие или выйти в главное меню.

Листинг кода данного модуля:

void ShowPositions()

{

ReadFile();

bool bExitFunc = false;

while (!bExitFunc)

{

cout << endl << "Введите одно из данных значений для поиска:" << endl;

cout << "1) введите 1 или |Show all| чтобы увидеть все товарные позиции в файле;" << endl;

cout << "2) введите 2 или |Show specific| чтобы увидеть определенную товарную позицию в файле." << endl;

//exit

string ChoiceInShowPos;

getline(cin, ChoiceInShowPos);

if (ChoiceInShowPos == "exit")

{

return;

}

if (ChoiceInShowPos == "Show all" || ChoiceInShowPos == "1")

{

//1) Show all

//красивый вывод всех позиций

for (int Index = 0; Index < TempFile.size(); Index++)

{

string TempOut = "";

for (int Jindex = 0; Jindex < 4; Jindex++)

{

TempOut += TempFile[Index][Jindex];

for (int SpaceCounter = 0; SpaceCounter < GetMaxColLength(Jindex) - TempFile[Index][Jindex].length() + 5; SpaceCounter++)

{

TempOut += " ";

}

}

cout << TempOut << endl;

}

bExitFunc = true;

}

else if (ChoiceInShowPos == "Show specific" || ChoiceInShowPos == "2")

{

//2) Show specific

//вывод определенной строки

//подсказка ввода кода товарa

bool bExitError = false;

int PositionByCode;

while (!bExitError)

{

cout << endl << "Введите код товара:" << endl;

//С этого момента данная переменая хранит код товара

getline(cin, ChoiceInShowPos);

if (ChoiceInShowPos == "exit")

{

return;

}

PositionByCode = stoi(FindPositionByCode(ChoiceInShowPos));

if (PositionByCode == -1)

{

//вывод ошибки

cout << endl << "ОШИБКА: код товара не найден в файле!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

}

else

{

bExitError = true;

}

}

//образование вывода

CoutSpecificPosition(PositionByCode);

bExitFunc = true;

}

else

{

//вывод ошибки и выбор повторить или меню

cout << endl << "ОШИБКА: введен неверный индекс действия!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

}

}

TempFile.clear();

}

Главный модуль RedactPositions() используется для редактирования позиций в файле через вектор временного хранения данных. В нем пользователь выбирает изменение кода, наименования, количества на складе и цены товара. В каждом из модулей происходит проверка на ошибку ввода, после которой пользователю дается возможность повтора ввода или выхода обратно в меню. Также перед изменением данных к пользователя спрашивается подтверждение изменения. Он может выбрать подтверждение изменений, повтор операции изменения или выход без изменения данных. Здесь также используется несколько дополнительных модулей, которые будут описаны ниже.

Листинг кода данного модуля:

void RedactPosition()

{

ReadFile();

//redact positions

cout << endl << "Введите код товара, данные которого вы хотите:" << endl;

string Code;

bool bExitCheck = false;

int Index = -1;

while (!bExitCheck)

{

getline(cin, Code);

if (Code == "exit")

{

return;

}

if (!bIsDigit(Code) || Code.size() != 5)

{

cout << endl << "ОШИБКА: код товара должен быть числом и состоять из пяти цифр!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

Index = stoi(FindPositionByCode(Code));

if (Index == -1)

{

//вывод ошибки

cout << endl << "ОШИБКА: код товара не найден в файле!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

else

{

bExitCheck = true;

}

}

bExitCheck = false;

while (!bExitCheck)

{

cout << endl << "Найдена позиция:" << endl;

CoutSpecificPosition(Index);

cout << endl << "Введите столбец, который вы хотите изменить:" << endl;

cout << "1) введите |1| или |Edit code| чтобы изменить код товара;" << endl;

cout << "2) введите |2| или |Edit name| чтобы изменить наименование товара;" << endl;

cout << "3) введите |3| или |Edit stock| чтобы изменить количество товара на складе;" << endl;

cout << "4) введите |4| или |Edit price| чтобы изменить цену товара." << endl;

getline(cin, Code);

if (Code == "exit")

{

return;

}

bool bExit = false;

if (Code == "1" || Code == "Edit code")

{

while (!bExit)

{

cout << endl << "Введите новый код товара:" << endl;

string TempNewCode, TempLastCode;

getline(cin, TempNewCode);

if (TempNewCode.size() != 5)

{

cout << endl << "ОШИБКА: код товара должен состоять из пяти символов!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (!bIsDigit(TempNewCode))

{

cout << endl << "ОШИБКА: код товара должен быть числом!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (FindPositionByCode(TempNewCode) != "-1")

{

cout << endl << "ОШИБКА: данный код товара уже существует в файле!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempLastCode = TempFile[Index][0];

TempFile[Index][0] = TempNewCode;

CoutSpecificPosition(Index);

cout << endl << "Убедитесь в правильности ввода нового кода товара." << endl;

cout << "Напишите |Yes| если новый код товара удовлетворителкн, для повтора ввода введите |No|." << endl;

cout << "Или, если хотите выйти в главное меню без сохранения изменений, напишите |exit|." << endl;

string ChangeChoice;

getline(cin, ChangeChoice);

if (ChangeChoice == "exit")

{

TempFile[Index][2] = TempLastCode;

return;

}

if (ChangeChoice == "Yes")

{

bExit = true;

bExitCheck = true;

break;

}

//введено No

TempFile[Index][2] = TempLastCode;

}

}

else if (Code == "2" || Code == "Edit name")

{

while (!bExit)

{

cout << endl << "Введите новое наименование товара на английском языке:" << endl;

string TempNewName, TempLastName;

getline(cin, TempNewName);

if (TempNewName.size() == 0)

{

cout << endl << "ОШИБКА: наименование товара не может быть пустым!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempLastName = TempFile[Index][1];

TempFile[Index][1] = TempNewName;

CoutSpecificPosition(Index);

cout << endl << "Убедитесь в правильности ввода нового наименования." << endl;

cout << "Новое наименование товара удовлетворяет правильности набора?" << endl;

cout << "Напишите |Yes| если новое наименование удовлетворительно, для повтора ввода введите |No|." << endl;

cout << "Или, если хотите выйти в главное меню без сохранения изменений, напишите |exit|." << endl;

string ChangeChoice;

getline(cin, ChangeChoice);

if (ChangeChoice == "exit")

{

TempFile[Index][1] = TempLastName;

return;

}

if (ChangeChoice == "Yes")

{

bExit = true;

bExitCheck = true;

break;

}

//введено No

TempFile[Index][1] = TempLastName;

}

}

else if (Code == "3" || Code == "Edit stock")

{

while (!bExit)

{

cout << endl << "Введите новое количество товара на складе:" << endl;

string TempNewStock, TempLastStock;

getline(cin, TempNewStock);

if (TempNewStock.size() == 0)

{

cout << endl << "ОШИБКА: количество товара не может быть пустым!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (!bIsDigit(TempNewStock))

{

cout << endl << "ОШИБКА: количество товара должен быть числом!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempLastStock = TempFile[Index][2];

TempFile[Index][2] = TempNewStock;

CoutSpecificPosition(Index);

cout << endl << "Убедитесь в правильности ввода нового количества товара." << endl;

cout << "Напишите |Yes| если новое количество товаров удовлетворительно, для повтора ввода введите |No|." << endl;

cout << "Или, если хотите выйти в главное меню без сохранения изменений, напишите |exit|." << endl;

string ChangeChoice;

getline(cin, ChangeChoice);

if (ChangeChoice == "exit")

{

TempFile[Index][2] = TempLastStock;

return;

}

if (ChangeChoice == "Yes")

{

bExit = true;

bExitCheck = true;

break;

}

//введено No

TempFile[Index][2] = TempLastStock;

}

}

else if (Code == "4" || Code == "Edit price")

{

while (!bExit)

{

cout << endl << "Введите новую цену товара:" << endl;

string TempNewPrice, TempLastPrice;

getline(cin, TempNewPrice);

if (TempNewPrice.size() == 0)

{

cout << endl << "ОШИБКА: цена не может быть пустой!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

int DotCounter = 0;

bool bIsDigit = true;

for (int CheckIndex = 0; CheckIndex < TempNewPrice.size(); CheckIndex++)

{

if (TempNewPrice[CheckIndex] == ',' && DotCounter < 1)

{

DotCounter++;

continue;

}

if (!isdigit(TempNewPrice[CheckIndex]))

{

bIsDigit = false;

break;

}

}

if (!bIsDigit)

{

cout << endl << "ОШИБКА: цена товара должна быть числом и иметь максимум одну запятую!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempLastPrice = TempFile[Index][3];

TempFile[Index][3] = TempNewPrice;

CoutSpecificPosition(Index);

cout << endl << "Убедитесь в правильности ввода цены товара." << endl;

cout << "Напишите |Yes| если новая цена удовлетворительна, для повтора ввода введите |No|." << endl;

cout << "Или, если хотите выйти в главное меню без сохранения изменений, напишите |exit|." << endl;

string ChangeChoice;

getline(cin, ChangeChoice);

if (ChangeChoice == "exit")

{

TempFile[Index][3] = TempLastPrice;

return;

}

if (ChangeChoice == "Yes")

{

bExit = true;

bExitCheck = true;

break;

}

//введено No

TempFile[Index][3] = TempLastPrice;

}

}

else

{

//ошибка

//вывод ошибки и выбор повторить или меню

cout << endl << "ОШИБКА: введен неверный индекс действия!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

}

}

WriteFile();

TempFile.clear();

}

Главный модуль AddPosition() даёт пользователю возможность добавить в файл ещё одну строку с данными, которая включает в себя код, наименование, количество на складе и цену товара. Каждый ввод пользователя проверяется на корректность введения данных. При некорректном введении данных пользователю даётся возможность ввести данные заново или выйти в меню. Перед концом выполнения модуля пользователю выводят введённые данный, после которых он подтверждает добавление, отклоняет добавление с повторным введением данных или выходит из модуль без сохранения изменений. Содержит несколько дополнительных модулей, которые будут описаны ниже.

Листинг кода данного модуля:

void AddPosition()

{

ReadFile();

vector<string> TempStrColoumn{};

while (true)

{

while (true)

{

cout << endl << "Введите код товара для новой позиции. Он должен состоять из 5 цифр." << endl;

string NewCode;

getline(cin, NewCode);

if (NewCode == "exit")

{

return;

}

if (NewCode.size() != 5 || !bIsDigit(NewCode))

{

cout << endl << "ОШИБКА: код товара должен быть числом и состоять из пяти цифр!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (FindPositionByCode(NewCode) != "-1")

{

cout << endl << "ОШИБКА: данный код товара уже существует в файле!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempStrColoumn.push\_back(NewCode);

break;

}

while (true)

{

cout << endl << "Введите наименование товара на английском языке:" << endl;

string NewName;

getline(cin, NewName);

if (NewName == "exit")

{

return;

}

if (NewName.size() == 0)

{

cout << endl << "ОШИБКА: наименование товара не может быть пустым!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempStrColoumn.push\_back(NewName);

break;

}

while (true)

{

cout << endl << "Введите количество товара на складе:" << endl;

string NewStock;

getline(cin, NewStock);

if (NewStock == "exit")

{

return;

}

if (NewStock.size() == 0)

{

cout << endl << "ОШИБКА: количество товара не может быть пустым!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (!bIsDigit(NewStock))

{

cout << endl << "ОШИБКА: количество товара должен быть числом!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempStrColoumn.push\_back(NewStock);

break;

}

while (true)

{

cout << endl << "Введите цену товара:" << endl;

string NewPrice;

getline(cin, NewPrice);

if (NewPrice == "exit")

{

return;

}

if (NewPrice.size() == 0)

{

cout << endl << "ОШИБКА: цена не может быть пустой!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

int DotCounter = 0;

bool bIsDigit = true;

for (int CheckIndex = 0; CheckIndex < NewPrice.size(); CheckIndex++)

{

if (NewPrice[CheckIndex] == ',' && DotCounter < 1)

{

DotCounter++;

continue;

}

if (!isdigit(NewPrice[CheckIndex]))

{

bIsDigit = false;

break;

}

}

if (!bIsDigit)

{

cout << endl << "ОШИБКА: цена товара должна быть числом и иметь максимум одну запятую!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempStrColoumn.push\_back(NewPrice);

break;

}

TempFile.push\_back(TempStrColoumn);

cout << endl << "Вы ввели:" << endl;

CoutSpecificPosition(TempFile.size() - 1);

cout << endl << "Если есть ошибки, вы можете их исправить в функции редактирования позиций." << endl;

cout << "Можно повторить операцию или выйти без сохранения изменений." << endl;

cout << "Напишите |Yes| если новая позиция удовлетворительна, для повтора ввода введите |No|." << endl;

cout << "Или, если хотите выйти в главное меню без сохранения изменений, напишите |exit|." << endl;

string ChangeChoice;

getline(cin, ChangeChoice);

if (ChangeChoice == "exit")

{

return;

}

if (ChangeChoice == "No")

{

continue;

}

break;

}

WriteFile();

TempFile.clear();

}

Главный модуль DeletePosition() даёт возможность пользователю удалить выбранную строку. После начала работы запрашивается код товара. При несуществующем в файле коде даётся возможность пользователю ввести код заново или выйти в меню из данного модуля. При существующем коде выводится строка под удаление и запрашивается подтверждение. При вводе отрицательного ответа No или exit удаление не происходит и пользователь возвращается обратно в меню. При вводе положительного ответа Yes происходит удаление строки. В данном модуле имеется несколько дополнительных модулей.

Листинг кода данного модуля:

void DeletePosition()

{

ReadFile();

string Code;

while (true)

{

cout << endl << "Введите позицию которую надо удалить." << endl;

getline(cin, Code);

if (Code == "exit")

{

return;

}

if (!bIsDigit(Code) || Code.size() != 5)

{

cout << endl << "ОШИБКА: код товара должен быть числом и состоять из пяти цифр!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (FindPositionByCode(Code) == "-1")

{

cout << endl << "ОШИБКА: данный код товара не существует в файле!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

break;

}

cout << endl << "Вы уверены что хотите удалить эту позицию?" << endl;

CoutSpecificPosition(stoi(FindPositionByCode(Code)));

cout << endl << "Введите |Yes| если хотите удалить эту позицию или |No| если хотите вернуться в меню и не удалять позицию." << endl;

string Choice;

getline(cin, Choice);

if (Choice == "No")

{

return;

}

TempFile.erase(TempFile.begin() + stoi(FindPositionByCode(Code)));

WriteFile();

}

Дополнительный модуль ReadFile() читает файл и вводит данные во временный файл, состоящий из вектора с вложенным вектором из строчного типа данных. Оптимизирует код и делает его читабельным путём использования одной строки кода в каждом главном модуле вместо бесконечного повторения одних и тех же строк.

Листинг кода данного модуля:

void ReadFile()

{

TempFile.clear();

File.open("ShopKeeperPositions.txt");

string FileLine;

char Delimiter = '/';

TempFile.push\_back({ "Code","Name","Stock","Price" });

while (getline(File, FileLine))

{

int Counter = 0; //считает колонки от 0 до 3

vector<string> TempStrColoumn{}; //Колонки

size\_t DelimiterPosition = 0;

while (/\*DelimiterPosition =\*/ FileLine.find(Delimiter) != string::npos)

{

DelimiterPosition = FileLine.find(Delimiter);

TempStrColoumn.push\_back(string(FileLine.substr(0, DelimiterPosition)));

FileLine.erase(0, DelimiterPosition + 1); //2 - длина делимитера

Counter++;

}

TempFile.push\_back(TempStrColoumn);

}

File.close();

}

Дополнительный модуль WriteFile() записывает данные в хранимый на компьютере файл. Берет данные из вектора со вложенным вектором из строчного типа данных. Разные столбцы разделяются ключевым символом «/». Данный модуль используется в трех главных модулях и помогает оптимизировать код путём использования там одной строки.

Листинг кода данного модуля:

void WriteFile()

{

ofstream oFile;

oFile.open("ShopKeeperPositions.txt");

for (int Index = 1; Index < TempFile.size(); Index++)

{

for (int Jindex = 0; Jindex < 4; Jindex++)

{

oFile << TempFile[Index][Jindex] << "/";

}

oFile << "\n";

}

oFile.close();

}

Дополнительный модуль CoutSpecificPosition() нужен для вывода определённой строки. Модуль вызывается в трёх главных модулях.

Листинг кода данного модуля:

void CoutSpecificPosition(int Code)

{

string TempOut = "";

for (int Jindex = 0; Jindex < 4; Jindex++)

{

//0 для заголовков

TempOut += TempFile[0][Jindex];

for (int SpaceCounter = 0; SpaceCounter < GetMaxColLength(Jindex) - TempFile[0][Jindex].length() + 5; SpaceCounter++)

{

TempOut += " ";

}

}

cout << TempOut << endl;

TempOut = "";

for (int Jindex = 0; Jindex < 4; Jindex++)

{

TempOut += TempFile[Code][Jindex];

for (int SpaceCounter = 0; SpaceCounter < GetMaxColLength(Jindex) - TempFile[Code][Jindex].length() + 5; SpaceCounter++)

{

TempOut += " ";

}

}

cout << TempOut << endl;

}

Дополнительный модуль GetMaxColLength() нужен для удобного вывода строк с данными. Ищет максимальный размер каждого столбца. При помощи него формируется ровный удобный вывод.

Листинг кода данного модуля:

{

//Для выравнивания вывода

int MaxLen = 0;

//Находим максимальную длину столбца для красивого вывода.

for (int Index = 1; Index < TempFile.size(); Index++)

{

if (TempFile[Index][Jindex].length() > MaxLen) MaxLen = TempFile[Index][Jindex].length();

}

return MaxLen;

}

Дополнительный модуль FindPositionByCode() используется для поиска номера строки в векторе со вложенным вектором из строчного типа. Принимает номер кода товара и по нему ищет номер строки. Используется там, где нужно найти определенную строку.

Листинг кода данного модуля:

{

for (int Index = 1; Index < TempFile.size(); Index++)

{

if (TempFile[Index][0] == Code)

{

return to\_string(Index);

}

}

return "-1";

}Дополнительный модуль ReturnChoiceInMain() используется в модуле Main() для перевода выбора из строчного типа в целочисленный для корректной работы switch.

Листинг кода данного модуля:

int ReturnChoiceInMain(string InterIndex)

Р{

//String to int

if (InterIndex == "Show positions" || InterIndex == "1")

{

return 1;

}

if (InterIndex == "Redact positions" || InterIndex == "2")

{

return 2;

}

if (InterIndex == "Add positions" || InterIndex == "3")

{

return 3;

}

if (InterIndex == "Delete positions" || InterIndex == "4")

{

return 4;

}

return 0;

}

Дополнительный модуль bIsDigit() используется для проверки введённого значения на число там, где нужно, чтобы ввод имел численное представление. Возвращает правду, если ввод был числом, или ложь в противоположном случае.

Листинг кода данного модуля:

bool bIsDigit(string Input)

{

for (int CheckIndex = 0; CheckIndex < Input.size(); CheckIndex++)

{

if (!isdigit(Input[CheckIndex]))

{

return false;

}

}

return true;

}

2 Испытания программы

2.1 Корректировка программы и программной документации по результатам испытаний

Во время отладки программы было обнаружено небольшое количество ошибок. Некоторые из этих ошибок являлись операциями с не назначенными для этого переменными. Для исправления данных ошибок потребовались минимальные действия в виде замещения их на назначенные для операций переменные.

Была ошибка, связанная с нерабочей функцией отчистит файла для записи на чистый файл новых данных. Данная проблема была исправлена посредством создания новой переменной ofstream oFile вместо старой fstream File.

Были допущены ошибки с бесконечными циклами, допущенные по невнимательности. Исправление данных ошибок простое – создание условия для выхода из цикла.

Описание программы

Данная программа имеет файл заголовка с описанными в нем функциями для возможности соблюдения любого порядка их написания. В файле заголовка также включены файлы: <fstream> для возможности манипуляции с данными файла; <string> для возможности работы со строковым типом; <vector> для создания вектора со вложенным вектором строкового типа, который позволит шире манипулировать данными файла. Также в файле заголовка описаны функции: int main(), который требуется для работы любых программ; ShowPositions(), использующийся для вывода строк файла; RedactPosition(), использующийся для редактирования строк файла; AddPosition(), при помощи которого пользователь может добавить строки с данными в файл; DeletePosition(), которым пользователь может удалять строки с данными из файла; ReadFile(), который читает файл для главных модулей; WriteFile(), который записывает строки в файл в трех главных модулях; CoutSpecificPosition(int Code), который выводит определенную строку из файла по коду из первого столбца; GetMaxColLength(int Index), который находит максимальную длину каждого столбца для удобного форматирования вывода данных пользователю; FindPositionByCode(string code), при помощи которого производится поиск определенной строки по коду из первого столбца; ReturnChoiceInMain(), обрабатывающий запрос пользователя в модуле Main(); проверяющий ввод пользователя на наличие там только числа. Дополнительно тут описаны две переменные: vector<vector<string>> TempFile{}, который хранит данные из файла для и во время манипуляций с ними; fstream File для хранения класса файла.

Главный файл с кодом в первых строках импортирует в себя файл заголовка для корректной работы программы. Также импортированы данные библиотеки: <iostream>, содержащий объекты, управляющие чтением из стандартных потоков и записью в них; <stdlib.h>, определяющий несколько функций общего назначения, в том числе - функции динамического управления памятью, генерации случайных чисел, целой арифметики, поиска, сортировки и преобразования типов данных.; <Windows.h> для возможности использования русского языка в консоли. Также одна строка описывает использование пространства имен std для простого доступа к некоторым функциям. Далее идет подробное описание функций, на каждую из которых будет выделен отдельный абзац за исключением main() и ReturnChoiceInMain().

Модуль int main() является самой главной функцией данной программы. Изначально в ней описываются две строки для использования русского языка в консольном выводе. Далее идет приветствие и объявление строчной переменной InterIndex для хранения ввода индекса действия пользователя в данном модуле. После этого идет цикл while, выход из которого происходит при вводе в переменную InterIndex значения «exit» и из которого следует завершение работы программы. После этого происходит запасная отчистка переменных InterIndex, TempFile и File для будущих итераций. После совершения данных действий выводятся подсказки для пользователя, называемое «меню», и вводится значение в InterIndex. При вводе «exit» программа завершает свою работу. Иначе происходит проверка на ввод в ReturnChoiceInMain() на ввод индекса действия: при вводе «1» или «Show positions» программа переходит в модуль ShowPositions(); при вводе «2» или «Redact positions» программа переходит в модуль RedactPosition(); при вводе «3» или «Add positions» программа переходит в модуль AddPosition; при вводе «4» или «Delete positions» программа переходит в модуль DeletePosition(); иначе при вводе любого другого значения кроме описанных выше программа выдаст пользователю искусственную ошибку, после которого цикл while продолжится. Каждый из данных модулей и их дополнительных модулей будет описан ниже. Также при каждом вводе «exit» позволит выйти из модуля в главный модуль main(), или позволит завершить работу программы в модуле main().

Модуль ShowPositions() сразу после вызова вызывает дополнительный модуль ReadFile() для переноса данных из текстового файла в векторную переменную, хранящую строки файла. Далее начинается цикл while. В нем пользователю дается выбор на вывод всех позиций при вводе «1» или «Show all» или вывод специфичной позиции при вводе «2» или «Show specific». Вывод данных всех позиций из файла осуществляется при помощи циклов for, один из которых вложен. Также используется модуль GetMaxColLength() для конфигурации удобного вывода данных пользователю. Вывод данных определенной позиции из файла осуществляется изначально запросом от пользователя кода позиции, который проверяется на существование в модуле FindPositionByCode(). Если позиции существует, то выводится нужная строка при помощи модуля CoutSpecificPosition(). После каждого из этих выводов данный модуль заканчивается и переходит в главный модуль main().

Модуль RedactPosition() после вызова вызывает модуль ReadFile(). Далее вводится код товара, который проверяется на корректность ввода и на существование в файле. Затем выводится найденная позиция при помощи CoutSpecificPosition() и запрашивает у пользователя выбор дальнейших действий: при вводе «1» или «Edit code» пользователь переходит к изменению кода товара, в котором новый введенный код проверяется на корректность ввода, корректность размера и отсутствие такого же кода в файле, и в итоге код изменяется на новый; действий: при вводе «2» или «Edit name» пользователь переходит к изменению наименования товара, где введенное новое наименование проходит проверку на ненулевой размер и после заменяет старое наименование; при вводе «3» или «Edit stock» пользователь переходит к изменению количества товара на складе, в котором новое значение проверяется на нулевой размер, на наличие только числа в вводе и после проверок старое значение заменяется на новое; при вводе «4» или «Edit price» пользователь переходит к изменению цены товара, в котором новое значение проходит проверку на ненулевой размер, наличие только числа с максимум одной запятой в вводе, и после проверок заменяет старое значение на новое. Каждая проверка на только число в воде происходит при помощи дополнительного модуля bIsDigit() за исключением проверки новой цены, в котором код похож на данный дополнительный модуль, но также включает проверку на наличие максимум одной запятой. При каждом изменении пользователя запрашивают подтвердить корректность изменения данных: при вводе «Yes» производится изменение данных; при вводе «No» пользователь заново производит изменение выбранных данных; при вводе «exit» пользователь выходит в главный модуль без сохранения изменений. После совершения всех операций вызывается метод WriteFile().

Модуль AddPosition() начинается с чтения файла дополнительным модулем ReadFile(), после которого по очередям вводятся код, наименование, количество на складе и цена товара. Код проверяется на размер из пяти чисел, наличия только цифр во вводе и уникальности кода модулем FindPositionByCode(). Наименование проверяется на ненулевой размер. Количество на складе проверяется на ненулевой размер и наличия только числа во вводе. Цена проверяется на ненулевой размер, наличие только чисел во вводе и максимум наличия одной запятой. Проверка на наличия только числа в воде производится модулем bIsDigit(). В итоге введённые данные выводятся на консоль пользователю, где у пользователя есть выбор, осуществляющийся вводом: при введенном «Yes» позиция добавляется в файл вызовом модуля WriteFile(); при «No» осуществляется повтор данного модуля; при «exit» изменения не сохраняются и программа выходит в модуль Main().

Модуль DeletePosition() начинается с вызова модуля ReadFile(), после которого у пользователя запрашивается существующий код. При вводе существующего кода выводится данная позиция модулем CoutSpecificPosition() и у пользователя спрашивается подтверждение удаления позиции: ввод «Yes» продолжает удаление позиции с последующей записью в файл модулем WriteFile(); «No» или «exit» отменяет изменение и выходит из данного модуля в модуль Main().

Дополнительный модуль ReadFile() изначально отчищает вектор, хранящий данные файла, File открывает текстовый файл и начинает переносить данные в вектор в цикле for с вложенным циклом for, разделяя столбцы символом «/». В конце File удаляет все хранимые значения.

Дополнительный модуль WriteFile() открывает текстовый файл при помощи переменной oFile класса ofstream, отчищает текстовый файл от данных, и записывает заново из вектора при помощи цикла for c вложенным циклом for, добавляя специальный разделитель столбцов «/».

Дополнительный модуль CoutSpecificPosition() начинает с вывода названия столбцов а затем выводит позицию при помощи цикла с вложенным циклом for, формируя удобный вывод дополнительным модулем GetMaxColLength.

Дополнительный модуль GetMaxColLength() при помощи цикла for перебирает каждую строчку столбца на максимальное значение длины столбца.

Дополнительный модуль FindPositionByCode() перебирает каждую строчку первого столбца с кодом товара. При существующем коде выводит индекс строки, иначе выводит -1.

Дополнительный модуль bIsDigit() перебирает каждый символ строчной переменной и проверяет на цифру. При каждой строчной переменной, содержащей только цифры, выводит true. Иначе выводит false.

Листинг программы:

1) файл заголовка .h

#pragma once

//#include <codecvt>

#include <fstream>

#include <string>

#include <vector>

#ifndef ShopKeeper

#define ShopKeeper

int main();

void ShowPositions();

void RedactPosition();

void AddPosition();

void DeletePosition();

void ReadFile();

void WriteFile();

void CoutSpecificPosition(int Code);

int GetMaxColLength(int Index);

std::string FindPositionByCode(std::string Code);

int ReturnChoiceInMain(std::string InterIndex);

bool bIsDigit(std::string Input);

std::vector<std::vector<std::string>> TempFile{};

std::fstream File;

#endif

2) файл кода .cpp

#include "ShopKeeper.h"

//#include <conio.h>

#include <iostream>

#include <stdlib.h>

//Для русского языка

#include <Windows.h>

using namespace std;

int main()

{

//Для русского языка

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << endl << "Добро пожаловать в ShowKeeper!" << endl;

string InterIndex;

while (InterIndex != "exit")

{

//Подстраховка

File.close();

TempFile.clear();

InterIndex.clear();

cout << endl << "Введите индекс выбора действия или введите |exit| для выхода из прокраммы:" << endl;

cout << "1) ввежите 1 или |Show positions| чтобы перейти в окно вывода данных файла;" << endl;

cout << "2) введите 2 или |Redact positions| чтобы перейти в окно изменения позиции;" << endl;

cout << "3) введите 3 или |Add positions| чтобы перейти в окно добавления позиции;" << endl;

cout << "4) введите 4 или |Delete positions| чтобы перейти в окно удаления позиции." << endl;

cout << "Подсказка: ввод |exit| в любой функции кроме меню вернет вас обратно в меню." << endl;

getline(cin, InterIndex);

if (InterIndex == "exit")

{

return 0;

}

switch (ReturnChoiceInMain(InterIndex))

{

case 1:

{

ShowPositions();

break;

}

case 2:

{

RedactPosition();

break;

}

case 3:

{

AddPosition();

break;

}

case 4:

{

DeletePosition();

break;

}

default:

cout << endl << "ОШИБКА: неверный ввод! Введите индекс с 1 по 4 или указанные ключевые фразы!" << endl;

break;

}

};

}

void ShowPositions()

{

ReadFile();

bool bExitFunc = false;

while (!bExitFunc)

{

cout << endl << "Введите одно из данных значений для поиска:" << endl;

cout << "1) введите 1 или |Show all| чтобы увидеть все товарные позиции в файле;" << endl;

cout << "2) введите 2 или |Show specific| чтобы увидеть определенную товарную позицию в файле." << endl;

//exit

string ChoiceInShowPos;

getline(cin, ChoiceInShowPos);

if (ChoiceInShowPos == "exit")

{

return;

}

if (ChoiceInShowPos == "Show all" || ChoiceInShowPos == "1")

{

//1) Show all

//красивый вывод всех позиций

for (int Index = 0; Index < TempFile.size(); Index++)

{

string TempOut = "";

for (int Jindex = 0; Jindex < 4; Jindex++)

{

TempOut += TempFile[Index][Jindex];

for (int SpaceCounter = 0; SpaceCounter < GetMaxColLength(Jindex) - TempFile[Index][Jindex].length() + 5; SpaceCounter++)

{

TempOut += " ";

}

}

cout << TempOut << endl;

}

bExitFunc = true;

}

else if (ChoiceInShowPos == "Show specific" || ChoiceInShowPos == "2")

{

//2) Show specific

//вывод определенной строки

//подсказка ввода кода товарa

bool bExitError = false;

int PositionByCode;

while (!bExitError)

{

cout << endl << "Введите код товара:" << endl;

//С этого момента данная переменая хранит код товара

getline(cin, ChoiceInShowPos);

if (ChoiceInShowPos == "exit")

{

return;

}

PositionByCode = stoi(FindPositionByCode(ChoiceInShowPos));

if (PositionByCode == -1)

{

//вывод ошибки

cout << endl << "ОШИБКА: код товара не найден в файле!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

}

else

{

bExitError = true;

}

}

//образование вывода

CoutSpecificPosition(PositionByCode);

bExitFunc = true;

}

else

{

//вывод ошибки и выбор повторить или меню

cout << endl << "ОШИБКА: введен неверный индекс действия!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

}

}

TempFile.clear();

}

void RedactPosition()

{

ReadFile();

//redact positions

cout << endl << "Введите код товара, данные которого вы хотите:" << endl;

string Code;

bool bExitCheck = false;

int Index = -1;

while (!bExitCheck)

{

getline(cin, Code);

if (Code == "exit")

{

return;

}

if (!bIsDigit(Code) || Code.size() != 5)

{

cout << endl << "ОШИБКА: код товара должен быть числом и состоять из пяти цифр!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

Index = stoi(FindPositionByCode(Code));

if (Index == -1)

{

//вывод ошибки

cout << endl << "ОШИБКА: код товара не найден в файле!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

else

{

bExitCheck = true;

}

}

bExitCheck = false;

while (!bExitCheck)

{

cout << endl << "Найдена позиция:" << endl;

CoutSpecificPosition(Index);

cout << endl << "Введите столбец, который вы хотите изменить:" << endl;

cout << "1) введите |1| или |Edit code| чтобы изменить код товара;" << endl;

cout << "2) введите |2| или |Edit name| чтобы изменить наименование товара;" << endl;

cout << "3) введите |3| или |Edit stock| чтобы изменить количество товара на складе;" << endl;

cout << "4) введите |4| или |Edit price| чтобы изменить цену товара." << endl;

getline(cin, Code);

if (Code == "exit")

{

return;

}

bool bExit = false;

if (Code == "1" || Code == "Edit code")

{

while (!bExit)

{

cout << endl << "Введите новый код товара:" << endl;

string TempNewCode, TempLastCode;

getline(cin, TempNewCode);

if (TempNewCode.size() != 5)

{

cout << endl << "ОШИБКА: код товара должен состоять из пяти символов!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (!bIsDigit(TempNewCode))

{

cout << endl << "ОШИБКА: код товара должен быть числом!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (FindPositionByCode(TempNewCode) != "-1")

{

cout << endl << "ОШИБКА: данный код товара уже существует в файле!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempLastCode = TempFile[Index][0];

TempFile[Index][0] = TempNewCode;

CoutSpecificPosition(Index);

cout << endl << "Убедитесь в правильности ввода нового кода товара." << endl;

cout << "Напишите |Yes| если новый код товара удовлетворителкн, для повтора ввода введите |No|." << endl;

cout << "Или, если хотите выйти в главное меню без сохранения изменений, напишите |exit|." << endl;

string ChangeChoice;

getline(cin, ChangeChoice);

if (ChangeChoice == "exit")

{

TempFile[Index][2] = TempLastCode;

return;

}

if (ChangeChoice == "Yes")

{

bExit = true;

bExitCheck = true;

break;

}

//введено No

TempFile[Index][2] = TempLastCode;

}

}

else if (Code == "2" || Code == "Edit name")

{

while (!bExit)

{

cout << endl << "Введите новое наименование товара на английском языке:" << endl;

string TempNewName, TempLastName;

getline(cin, TempNewName);

if (TempNewName.size() == 0)

{

cout << endl << "ОШИБКА: наименование товара не может быть пустым!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempLastName = TempFile[Index][1];

TempFile[Index][1] = TempNewName;

CoutSpecificPosition(Index);

cout << endl << "Убедитесь в правильности ввода нового наименования." << endl;

cout << "Новое наименование товара удовлетворяет правильности набора?" << endl;

cout << "Напишите |Yes| если новое наименование удовлетворительно, для повтора ввода введите |No|." << endl;

cout << "Или, если хотите выйти в главное меню без сохранения изменений, напишите |exit|." << endl;

string ChangeChoice;

getline(cin, ChangeChoice);

if (ChangeChoice == "exit")

{

TempFile[Index][1] = TempLastName;

return;

}

if (ChangeChoice == "Yes")

{

bExit = true;

bExitCheck = true;

break;

}

//введено No

TempFile[Index][1] = TempLastName;

}

}

else if (Code == "3" || Code == "Edit stock")

{

while (!bExit)

{

cout << endl << "Введите новое количество товара на складе:" << endl;

string TempNewStock, TempLastStock;

getline(cin, TempNewStock);

if (TempNewStock.size() == 0)

{

cout << endl << "ОШИБКА: количество товара не может быть пустым!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (!bIsDigit(TempNewStock))

{

cout << endl << "ОШИБКА: количество товара должен быть числом!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempLastStock = TempFile[Index][2];

TempFile[Index][2] = TempNewStock;

CoutSpecificPosition(Index);

cout << endl << "Убедитесь в правильности ввода нового количества товара." << endl;

cout << "Напишите |Yes| если новое количество товаров удовлетворительно, для повтора ввода введите |No|." << endl;

cout << "Или, если хотите выйти в главное меню без сохранения изменений, напишите |exit|." << endl;

string ChangeChoice;

getline(cin, ChangeChoice);

if (ChangeChoice == "exit")

{

TempFile[Index][2] = TempLastStock;

return;

}

if (ChangeChoice == "Yes")

{

bExit = true;

bExitCheck = true;

break;

}

//введено No

TempFile[Index][2] = TempLastStock;

}

}

else if (Code == "4" || Code == "Edit price")

{

while (!bExit)

{

cout << endl << "Введите новую цену товара:" << endl;

string TempNewPrice, TempLastPrice;

getline(cin, TempNewPrice);

if (TempNewPrice.size() == 0)

{

cout << endl << "ОШИБКА: цена не может быть пустой!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

int DotCounter = 0;

bool bIsDigit = true;

for (int CheckIndex = 0; CheckIndex < TempNewPrice.size(); CheckIndex++)

{

if (TempNewPrice[CheckIndex] == ',' && DotCounter < 1)

{

DotCounter++;

continue;

}

if (!isdigit(TempNewPrice[CheckIndex]))

{

bIsDigit = false;

break;

}

}

if (!bIsDigit)

{

cout << endl << "ОШИБКА: цена товара должна быть числом и иметь максимум одну запятую!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempLastPrice = TempFile[Index][3];

TempFile[Index][3] = TempNewPrice;

CoutSpecificPosition(Index);

cout << endl << "Убедитесь в правильности ввода цены товара." << endl;

cout << "Напишите |Yes| если новая цена удовлетворительна, для повтора ввода введите |No|." << endl;

cout << "Или, если хотите выйти в главное меню без сохранения изменений, напишите |exit|." << endl;

string ChangeChoice;

getline(cin, ChangeChoice);

if (ChangeChoice == "exit")

{

TempFile[Index][3] = TempLastPrice;

return;

}

if (ChangeChoice == "Yes")

{

bExit = true;

bExitCheck = true;

break;

}

//введено No

TempFile[Index][3] = TempLastPrice;

}

}

else

{

//ошибка

//вывод ошибки и выбор повторить или меню

cout << endl << "ОШИБКА: введен неверный индекс действия!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

}

}

WriteFile();

TempFile.clear();

}

void AddPosition()

{

ReadFile();

vector<string> TempStrColoumn{};

while (true)

{

while (true)

{

cout << endl << "Введите код товара для новой позиции. Он должен состоять из 5 цифр." << endl;

string NewCode;

getline(cin, NewCode);

if (NewCode == "exit")

{

return;

}

if (NewCode.size() != 5 || !bIsDigit(NewCode))

{

cout << endl << "ОШИБКА: код товара должен быть числом и состоять из пяти цифр!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (FindPositionByCode(NewCode) != "-1")

{

cout << endl << "ОШИБКА: данный код товара уже существует в файле!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempStrColoumn.push\_back(NewCode);

break;

}

while (true)

{

cout << endl << "Введите наименование товара на английском языке:" << endl;

string NewName;

getline(cin, NewName);

if (NewName == "exit")

{

return;

}

if (NewName.size() == 0)

{

cout << endl << "ОШИБКА: наименование товара не может быть пустым!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempStrColoumn.push\_back(NewName);

break;

}

while (true)

{

cout << endl << "Введите количество товара на складе:" << endl;

string NewStock;

getline(cin, NewStock);

if (NewStock == "exit")

{

return;

}

if (NewStock.size() == 0)

{

cout << endl << "ОШИБКА: количество товара не может быть пустым!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (!bIsDigit(NewStock))

{

cout << endl << "ОШИБКА: количество товара должен быть числом!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempStrColoumn.push\_back(NewStock);

break;

}

while (true)

{

cout << endl << "Введите цену товара:" << endl;

string NewPrice;

getline(cin, NewPrice);

if (NewPrice == "exit")

{

return;

}

if (NewPrice.size() == 0)

{

cout << endl << "ОШИБКА: цена не может быть пустой!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

int DotCounter = 0;

bool bIsDigit = true;

for (int CheckIndex = 0; CheckIndex < NewPrice.size(); CheckIndex++)

{

if (NewPrice[CheckIndex] == ',' && DotCounter < 1)

{

DotCounter++;

continue;

}

if (!isdigit(NewPrice[CheckIndex]))

{

bIsDigit = false;

break;

}

}

if (!bIsDigit)

{

cout << endl << "ОШИБКА: цена товара должна быть числом и иметь максимум одну запятую!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

TempStrColoumn.push\_back(NewPrice);

break;

}

TempFile.push\_back(TempStrColoumn);

cout << endl << "Вы ввели:" << endl;

CoutSpecificPosition(TempFile.size() - 1);

cout << endl << "Если есть ошибки, вы можете их исправить в функции редактирования позиций." << endl;

cout << "Можно повторить операцию или выйти без сохранения изменений." << endl;

cout << "Напишите |Yes| если новая позиция удовлетворительна, для повтора ввода введите |No|." << endl;

cout << "Или, если хотите выйти в главное меню без сохранения изменений, напишите |exit|." << endl;

string ChangeChoice;

getline(cin, ChangeChoice);

if (ChangeChoice == "exit")

{

return;

}

if (ChangeChoice == "No")

{

continue;

}

break;

}

WriteFile();

TempFile.clear();

}

void DeletePosition()

{

ReadFile();

string Code;

while (true)

{

cout << endl << "Введите позицию которую надо удалить." << endl;

getline(cin, Code);

if (Code == "exit")

{

return;

}

if (!bIsDigit(Code) || Code.size() != 5)

{

cout << endl << "ОШИБКА: код товара должен быть числом и состоять из пяти цифр!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

if (FindPositionByCode(Code) == "-1")

{

cout << endl << "ОШИБКА: данный код товара не существует в файле!" << endl;

cout << "Вы хотите повторить операцию? Напишите |Yes| если хотите повторить или |No| если хотите выйти в главное меню." << endl;

string ErrorChoice;

getline(cin, ErrorChoice);

if (ErrorChoice == "No" || ErrorChoice == "Exit")

{

return;

}

continue;

}

break;

}

cout << endl << "Вы уверены что хотите удалить эту позицию?" << endl;

CoutSpecificPosition(stoi(FindPositionByCode(Code)));

cout << endl << "Введите |Yes| если хотите удалить эту позицию или |No| если хотите вернуться в меню и не удалять позицию." << endl;

string Choice;

getline(cin, Choice);

if (Choice == "No" || Choice == "exit")

{

return;

}

TempFile.erase(TempFile.begin() + stoi(FindPositionByCode(Code)));

WriteFile();

}

void ReadFile()

{

TempFile.clear();

File.open("ShopKeeperPositions.txt");

string FileLine;

char Delimiter = '/';

TempFile.push\_back({ "Code","Name","Stock","Price" });

while (getline(File, FileLine))

{

int Counter = 0; //считает колонки от 0 до 3

vector<string> TempStrColoumn{}; //Колонки

size\_t DelimiterPosition = 0;

while (/\*DelimiterPosition =\*/ FileLine.find(Delimiter) != string::npos)

{

DelimiterPosition = FileLine.find(Delimiter);

TempStrColoumn.push\_back(string(FileLine.substr(0, DelimiterPosition)));

FileLine.erase(0, DelimiterPosition + 1); //2 - длина делимитера

Counter++;

}

TempFile.push\_back(TempStrColoumn);

}

File.close();

}

void WriteFile()

{

ofstream oFile;

oFile.open("ShopKeeperPositions.txt");

for (int Index = 1; Index < TempFile.size(); Index++)

{

for (int Jindex = 0; Jindex < 4; Jindex++)

{

oFile << TempFile[Index][Jindex] << "/";

}

oFile << "\n";

}

oFile.close();

}

void CoutSpecificPosition(int Code)

{

string TempOut = "";

for (int Jindex = 0; Jindex < 4; Jindex++)

{

//0 для заголовков

TempOut += TempFile[0][Jindex];

for (int SpaceCounter = 0; SpaceCounter < GetMaxColLength(Jindex) - TempFile[0][Jindex].length() + 5; SpaceCounter++)

{

TempOut += " ";

}

}

cout << TempOut << endl;

TempOut = "";

for (int Jindex = 0; Jindex < 4; Jindex++)

{

TempOut += TempFile[Code][Jindex];

for (int SpaceCounter = 0; SpaceCounter < GetMaxColLength(Jindex) - TempFile[Code][Jindex].length() + 5; SpaceCounter++)

{

TempOut += " ";

}

}

cout << TempOut << endl;

}

int GetMaxColLength(int Jindex)

{

//Для выравнивания вывода

int MaxLen = 0;

//Находим максимальную длину столбца для красивого вывода.

for (int Index = 1; Index < TempFile.size(); Index++)

{

if (TempFile[Index][Jindex].length() > MaxLen) MaxLen = TempFile[Index][Jindex].length();

}

return MaxLen;

}

string FindPositionByCode(string Code)

{

for (int Index = 1; Index < TempFile.size(); Index++)

{

if (TempFile[Index][0] == Code)

{

return to\_string(Index);

}

}

return "-1";

}

int ReturnChoiceInMain(string InterIndex)

{

//String to int

if (InterIndex == "Show positions" || InterIndex == "1")

{

return 1;

}

if (InterIndex == "Redact positions" || InterIndex == "2")

{

return 2;

}

if (InterIndex == "Add positions" || InterIndex == "3")

{

return 3;

}

if (InterIndex == "Delete positions" || InterIndex == "4")

{

return 4;

}

return 0;

}

bool bIsDigit(string Input)

{

for (int CheckIndex = 0; CheckIndex < Input.size(); CheckIndex++)

{

if (!isdigit(Input[CheckIndex]))

{

return false;

}

}

return true;

}

Инструкция по эксплуатации программы

1 Главное меню программы

Для запуска программы откройте папку ShopKeeper и запустите ShopKeeper.exe. Если приложения не было в папке ShopKeeper, то откройте в этой папке папку Debug и оттуда запустите приложение ShopKeeper.exe.

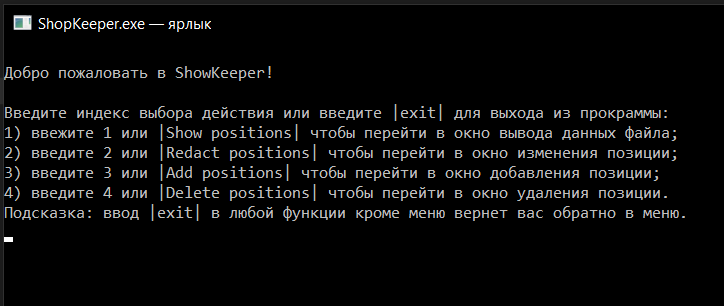


Рис.1 – главное меню.

После запуска приложение вы увидите меню как на рисунке № 1. Здесь описаны все функции данного приложения. Для выхода из программы введите «exit».

Для того, чтобы посмотреть все позиции товаров или увидеть определенную позицию товара, то введите «1» или «Show position». После ввода вы перейдете в меню вывода позиций.

Для того, чтоб изменить определенные данные определенной позиции, то введите «2» или «Redact positions». После ввода вы перейдете в меню изменения позиций.

Для добавления позиций введите «3» или «Add positions». После ввода вы перейдете в меню добавления позиций.

Для удаления позиций введите «4» или «Delete positions». После ввода вы перейдете в меню удаления позиций.

Не забудьте, что при вводе в консоль «exit» вы выйдете в главное меню из любого другого меню. Если вы введете «exit» в главном меню, то вы завершите работу программы.

2 Меню вывода позиций

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2 – меню вывода позиций.

После перехода в данное меню вы увидите в консоли тот же вывод, что и на рисунке № 2. Вам предлагается на выбор вывод всех позиций или вывод определенной позиции. Для вывода всех позиций введите «1» или «Show all». Для вывода определенной позиции введите «2» или «Show specific». Введите «exit» для выхода в главное меню.

2.1 Вывод всех позиций

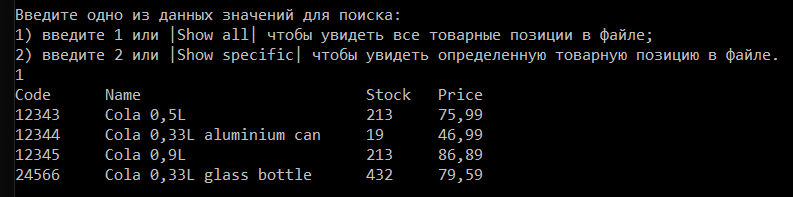


Рис.3 - пример вывода всех позиций.

После выбора вывода всех позиций программа должная вывести все позиции, существующие в программе. Пример данного вывода продемонстрирован на рисунке № 3.

2.2 вывод определенной позиции

Изображение выглядит как текст

Автоматически созданное описание

Рис. 4 - пример вывода определенной позиции.

После выбора вывода определенной позиции вы должны будете ввести код товара. Код товара всегда будет являться пятизначным числом. Далее вам выведется все данные определенной позиции.

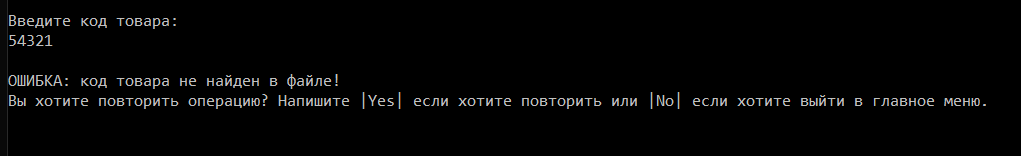


Рис. 5 - пример ошибки.

Если вы ввели несуществующий код товара, то программа вам выдаст ошибку. Вы сможете повторить операцию, введя «Yes», или выйти в главное меню, введя «No».

3 Меню изменения позиции



Рис. 6 – меню изменения позиции.

После перехода в данное меню вас попросят ввести код товара, который вы хотите изменить. При ошибке вас попросят ввести код заново или выйти из этого меню. При успешном вводе кода вам выведется данные по позиции с данным кодом и будет доступен выбор изменения.

Для изменения кода товара введите «1» или «Edit code». Вы перейдете в состояние изменения кода.

Для изменения наименования товара введите «2» или «Edit name». Вы перейдете в состояние изменения наименования.

Для изменения количества товара введите «3» или «Edit stock». Вы перейдете в состояние изменения количества товара.

Для изменения цены товара введите «4» или «Edit price». Вы перейдете в состояние изменения цены.

3.1 Изменение кода товара

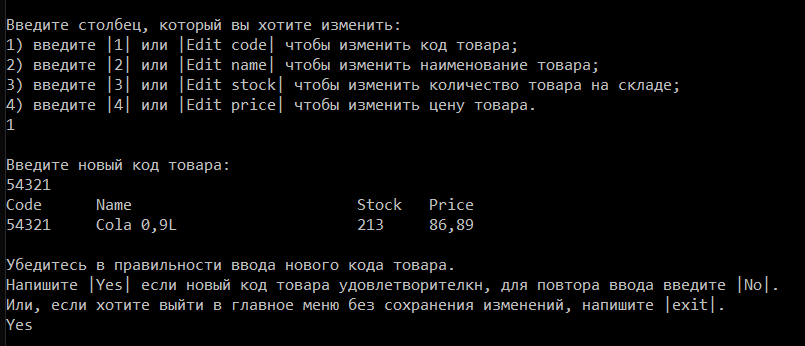


Рис. 7 – изменение кода товара.

После перехода в состояние изменения кода товара вас попросят ввести новый код товара. На ввод принимается строго число из пяти цифр. При какой-либо ошибке вы сможете повторить операцию или выйти в меню. После ввода вас попросят подтвердить данное изменение. Введите «Yes» если вы подтверждаете данное изменение, «No», если хотите ввести код еще раз, «exit», если вы хотите выйти без сохранения изменений. Пример вы можете увидеть на рисунке № 7.

3.2 Изменение наименования товара

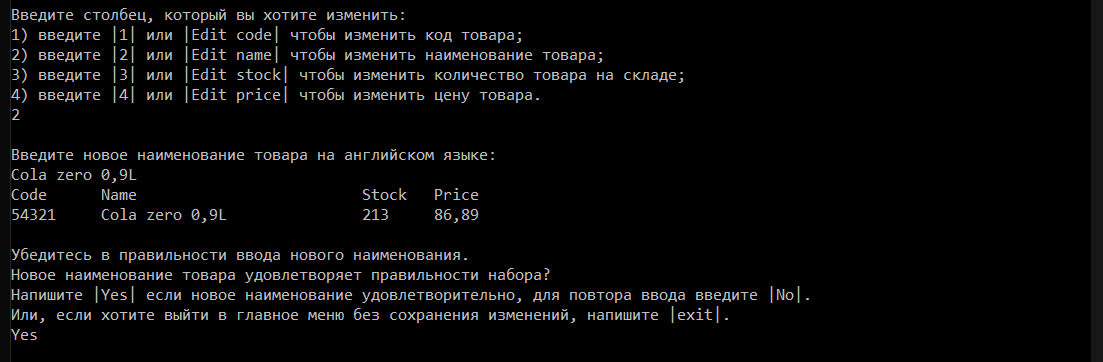


Рис. 8 – изменение наименования товара.

После перехода в состояние изменения наименования товара вас попросят ввести новое наименования товара. Вводите только на английском языке! При какой-либо ошибке вы сможете повторить операцию или выйти в меню. После ввода вас попросят подтвердить данное изменение. Введите «Yes» если вы подтверждаете данное изменение, «No», если хотите ввести наименование еще раз, «exit», если вы хотите выйти без сохранения изменений. Пример вы можете увидеть на рисунке № 8.

3.3 Изменение количества товара

Изображение выглядит как текст

Автоматически созданное описание

Рис.9 – изменение количества товара.

После перехода в состояние изменения количества товара вас попросят ввести новое количество товара. Вводите только целое число! При какой-либо ошибке вы сможете повторить операцию или выйти в меню. После ввода вас попросят подтвердить данное изменение. Введите «Yes» если вы подтверждаете данное изменение, «No», если хотите ввести количество товара еще раз, «exit», если вы хотите выйти без сохранения изменений. Пример вы можете увидеть на рисунке № 9.

3.4 Изменение цены товара

Изображение выглядит как текст

Автоматически созданное описание

Рис. 10 – изменение цены товара.

После перехода в состояние изменения цены товара вас попросят ввести новую цену товара. Вводите только число с одной запятой! При какой-либо ошибке вы сможете повторить операцию или выйти в меню. После ввода вас попросят подтвердить данное изменение. Введите «Yes» если вы подтверждаете данное изменение, «No», если хотите ввести цену товара еще раз, «exit», если вы хотите выйти без сохранения изменений. Пример вы можете увидеть на рисунке № 10.

4 Меню добавления позиции