Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

[Институт прикладных информационных технологий](http://rasp.sstu.ru/) и коммуникаций

Кафедра Информационная безопасность автоматизированных систем

Направление подготовки 10.03.01 Информационная безопасность

**Расчётно-графическая работа**

**по дисциплине «Языки программирования»**

**«Складское приложение»**

Выполнил: студент 1 курса

учебной группы с-ИБС11

очной формы обучения

Володов Никита Дмитриевич

Проверил: ассистент каф. ИБС

Романчук С. П.

Саратов 2020

**Аннотация к работе**

**Складское приложение**

Данная компьютерная программа может выполнять функции приложения складского учёта товаров. Она должна реализовывать подсчёты количества имеющегося товара и оборудования на потенциальном заводском складе.

Программа должна обладать следующими возможностями:

• создание, удаление и редактирование операций поступления и реализации товаров

• просмотр остатков на складе на определенную дату

• просмотр истории операций и поиск операций по различным характеристикам,

• загрузка и сохранение истории операций в файл.

При разработке приложения обязательно должены быть реализованы классы, моделирующие операцию поступление товара, операцию реализации товара, контрагент, склад и товар.

Оглавление

[Введение 4](#_Toc43377487)

[Тема и постановка задачи 5](#_Toc43377487)

[Теоретическая часть 6](#_Toc43377487)

[Практическая часть 8](#_Toc43377487)

[Заключение 10](#_Toc43377487)

[Приложения 11](#_Toc43377487)

[Список используемой литературы (ссылок) 54](#_Toc43377487)

# Введение

Большинству сотрудников в розничной торговле свойственно безразлично относится к покупателям, особенно, если нет должного контроля со стороны руководителя. Но наблюдать за деятельность персонала может и [программа для ведения клиентской базы](https://www.ekam.ru/page/programma-skladskogo-ucheta), которая не даст уклониться от разговора, звонка или консультации.

Для примера можно рассмотреть две пиццерии, предлагающие доставку на дом и принимающие заказы, в том числе, из соц-сетей. В одной из них установлена автоматическая система взаимодействия с клиентами (CRM-система), а в другой нет.

В первом случае заказ из социальной сети будет автоматически принят программой, о чем менеджер получит уведомление. После этого сотрудник сразу позвонит клиенту, уточнит детали и успешно продаст товар. При этом руководитель увидит время, прошедшее от момента заказа до времени звонка клиенту.

Во втором случае менеджер может заходить в соц-сети всего несколько раз в день, довольствуясь заказами по телефону. А пропущенные в интернете заявки сотрудник имеет возможность просто удалять или аргументировать их невыполнение сильной занятостью. В результате уровень продаж в неавтоматизированном бизнесе будет ниже, да и клиенты, недовольные онлайн-сервисом, вряд ли придут в саму пиццерию.

Таким образом, правильно настроенная CRM-система позволяет наращивать количество лояльных покупателей и увеличивать повторные продажи. Но следует помнить, что сама по себе [программа для ведения клиентской базы](https://www.ekam.ru/page/programma-skladskogo-ucheta) работу не сделает, и конечный результат всегда зависит от умений персонала.

**Тема расчетно-графической работы:** «Складское приложение»

**Целью расчетно-графической работы** – улучшение навыков работы с интегрированной средой разработки **Code::Blocks** , получение теоретических знаний в области программирования на языке C++. А помимо этого получение навыка составления пояснительной записки и оформления в соответствии с требованиями, принятыми на направлении «Информационная безопасность автоматизированных систем».

**Задачами расчетно-графической работы являются:**

1) разработка программы на языке С++ в соответствии с заданием;

2) составление пояснительной записки в соответствии с требованиями.

**Законченный вариант приложения должен содержать:**

* Графическую оболочку с кнопками, полями ввода и таблицами, позволяющими анализировать и обрабатывать данные о товарах;
* Базу данных товаров, для хранения информации в долгосрочном формате;

# Теоретическая часть

**Краткое описание языка**

C++ — компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как

* процедурное программирование,
* объектно-ориентированное программирование,
* обобщённое программирование.

Ключевые особенности языка C++:

1. Полностью совместим с языком C;

2. Поддерживает разные стили программирования;

3. Является кроссплатформенным языком;

4. Строго типизирован;

5. Компилируемый;

6. Содержит инструменты для работы с шаблонами;

7. Работает со статической и динамической памятью;

8. Имеет возможность перегрузки операторов;

**Описание среды разработки**

**Code::Blocks** — свободная кроссплатформенная среда разработки. Code::Blocks написана на С++ и использует библиотеку wxWidgets. Имея открытую архитектуру, может масштабироваться за счёт подключаемых модулей. Поддерживает языки программирования С, С++, D (с ограничениями), Fortran.

Code::Blocks разрабатывается для Windows, Linux и Mac OS X. Среду можно собрать из исходников практически под любую Unix-подобную систему, например FreeBSD, PC-BSD.

В данной работе используется **GNU Compiler Collection.**

**GNU Compiler Collection** (обычно используется сокращение **GCC**) — набор компиляторов для различных языков программирования, разработанный в рамках проекта GNU. GCC является свободным программным обеспечением, распространяется фондом свободного программного обеспечения (FSF) на условиях GNU GPL и GNU LGPL и является ключевым компонентом GNU toolchain. Он используется как стандартный компилятор для свободных UNIX-подобных операционных систем.

Изначально названный **GNU C Compiler** поддерживал только язык Си. Позднее GCC был расширен для компиляции исходных кодов на таких языках программирования, как C++, Objective-C, Java, Фортран, Ada, Go, GAS и D.

# 

**Практическая часть**

Складское приложение в своём функционале подразумевает частое взаимодействие с вводом данных различного типа, их быстрой обработкой и сортировкой. В данной расчётно-графической работе эти возможности реализуются при помощи графических элементов фреймворка .NET Framework с интегрированной в него поддержкой языка C++.

Алгоритм работы:

**1.** Загружаем данные из файла. (**Если мы хотим создать новую базу данных, пункт 1.Загрузка данных из файла не используем (в этом случае старая база данных будет удалена!)**) **2.** Происходит инициализация загруженных данных.  
**3.** Далее мы можем выполнять различные действия с базой данных: просматривать товары со склада, добавить, изменить или удалить товар, либо весь список целиком, а также реализован функционал продажи товара. **4.**  Также мы можем выполнять поиск по различным характеристикам.  
**5.**  По завершению работы, сохраняем данные в файл и выходим из программы.

**Используемые библиотеки:  
  
1. #include <iostream> //** заголовочный файл библиотеки для работы с потоками ввода-вывода консоли

**2. #include <locale> //** заголовочный файл библиотеки для работы с локалями

**3. #include <vector> //** заголовочный файл библиотеки для работы с векторами

**4. #include <fstream> //** заголовочный файл библиотеки для работы с файловыми потоками

**5. #include <sstream> //** заголовочный файл библиотеки для работы с текстовыми потоками

**6. #include <windows.h> //** заголовочный файл библиотеки для работы с WINAPI  
А также, макрос отчистки потока ввода **- #define CIN\_FLUSH if (cin.peek()) cin.ignore()**

**В программе реализованы такие классы, как:**

* GoodsReceipt;
* SalesGoods;
* WareHouse;
* CounterParty;
* Products.

# Заключение

В ходе выполнения расчётно-графической работы было выполнено задание в соответствии с номером варианта, которое включало в себя составление рабочей программы на языке программирования C++. В результате  были выполнены следующие этапы:

1. Обследование и разработка технического задания.
2. Построение алгоритма программы.
3. Разработка программы.
4. Тестирование и устранение ошибок.

В процессе написания и отладки программ были закреплены навыки работы с разными возможностями языка C++. Итоговое приложение было протестировано, и результат проверки дал положительный результат. Исходный код приложения подробно прокомментирован, что позволяет без труда понять, изменить или дополнить часть его функционала.

 Таким образом, мы приходим к следующему выводу - использование разработанной программы  заметно  сократит время поиска  необходимых данных.

**Приложения**

**Код программы**

#include <iostream> // заголовочный файл библиотеки для работы с потоками ввода-вывода консоли.

#include <locale> // заголовочный файл стандартной библиотеки языка программирования С, который используется для задач, связанных с локализацией.

#include <vector> // заголовочный файл библиотеки для работы с векторами: Стандартный шаблон обобщённого программирования языка C++ std::vector<T> — реализация динамического массива.

#include <fstream> // заголовочный файл библиотеки для работы с файловыми потоками.

#include <sstream> // заголовочный файл библиотеки для работы с текстовыми потоками.

#include <windows.h> // заголовочный файл библиотеки для работы с WINAPI.

#define CIN\_FLUSH if (cin.peek()) cin.ignore() // макрос очистки потока ввода.

using namespace std; //пространство стандартных имён.

#include"GoodsReceipt.h" //классы с методами и переменными, которые мы создали для работы с ними.

#include"SalesGoods.h"

#include"WareHouse.h"

#include"CounterParty.h"

#include"Products.h"

void menu() { //Реализация удобного меню программы.

cout << "#########################################################################################" << endl;

cout << "################################## СКЛАДСКОЕ ПРИЛОЖЕНИЕ #################################" << endl;

cout << "#########################################################################################" << endl;

cout << "##################################### МЕНЮ ПРОГРАММЫ ####################################" << endl;

cout << "#########################################################################################" << endl;

cout << " ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

cout << " ~~~~~~~~~~~~~~~~~~~ Склад ~~~~~~~~~~~~~~~~~" << endl;

cout << " ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

cout << " |> 1. Поступление товара |" << endl;

cout << " |> 2. Посмотреть склад |" << endl;

cout << " |> 3. Исправить ошибку на складе |" << endl;

cout << " |> 4. Удалить Определённый товар |" << endl;

cout << " |> 5. Очистить склад |" << endl;

cout << " |> 6. Поиск товара по дате |" << endl;

cout << " |> 7. Поиск товара по названию |" << endl;

cout << " |> 8. Поиск товара по цене |" << endl;

cout << " |> 9. Поиск товара по количеству |" << endl;

cout << " |> 10. Загрузить данные из файла |" << endl;

cout << " |> 11. Сохранить данные в файл |" << endl;

cout << " ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

cout << " ~~~~~~~~~~~ Журнал учёта продаж ~~~~~~~~~~~" << endl;

cout << " ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

cout << " |> 12. Продажа товара |" << endl;

cout << " |> 13. Список продаж |" << endl;

cout << " |> 14. Редактировать список продаж |" << endl;

cout << " |> 15. Исключить проданный товар |" << endl;

cout << " |> 16. Очистить список продаж |" << endl;

cout << " |> 17. Поиск продаж товара по дате |" << endl;

cout << " |> 18. Поиск продаж товара по названию |" << endl;

cout << " |> 19. Поиск продаж товара по цене |" << endl;

cout << " |> 20. Поиск продаж товара по фирме |" << endl;

cout << " |> 21. Поиск продаж товара по количеству |" << endl;

cout << " |> 22. Загрузить данные из файла |" << endl;

cout << " |> 23. Сохранить данные в файл |" << endl;

cout << " ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

cout << " |> 0. Выход из программы |" << endl;

cout << " ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

cout << " Введите пукнт меню > ";

}

vector <WareHouse> Storage; //создание вектора для хранения товаров на складе.

vector <Products> Sales; //создание вектора для хранения проданных товаров.

void addProduct() //функция добавляет на склад новый товар.

{

system("cls"); //очистка консоли.

string x; //переменная для получения данных о товаре из консоли.

WareHouse c; //объект для создания нового товара.

CIN\_FLUSH; //очистка потока ввода.

cout << "Новая поставка..." << endl << endl;

cout << "Дата: "; getline(cin, x);

c.setDate(x);

cout << "товар: "; getline(cin, x);

c.setProduct(x);

cout << "Цена: "; getline(cin, x);

c.setPrice(x);

cout << "Количество: "; getline(cin, x);

c.setAmount(x);

Storage.push\_back(c);

cout << endl << "Поставка завершена успешно..." << endl << endl;

}

void printProduct() //функция вывода списка всех товаров на складе.

{

system("cls");

int num = 0; //переменная для нумерации контактов.

if(Storage.size()) //условие, показывающее, есть ли на складе товары.

{

cout << "=========================================================================================" << endl;

cout << "= №\tДата\t\tПродукт\t\tЦена\t\tКоличество\t=================" << endl;

cout << "=========================================================================================" << endl;

for (WareHouse c : Storage) //Благодаря циклу "для", проходим по все товарам на складе.

{

cout << "= " << (++num) << ".\t";

cout << c.getDate() << "\t";

if(c.getDate().length() < 8) cout << "\t";

cout << c.getProduct() << "\t";

if(c.getProduct().length() < 8) cout << "\t";

cout << c.getPrice() << "\t";

if(c.getPrice().length() < 8) cout << "\t";

cout << c.getAmount() << "\t";

if(c.getAmount().length() < 8) cout << "\t";

cout << "=================" << endl;

}

cout << "=========================================================================================" << endl << endl;

}

else

{

cout << "На складе пусто..." << endl;

}

cout << endl;

}

void fixProduct() //функция редактирует информацию о товарах на складе.

{

system("cls");

int num; //редактируемый товар со склада.

WareHouse c;

string x;

if(Storage.size())

{

printProduct();

cout << "Введите номер товара, который хотите отредактировать: ";

cin >> num;

cout << endl;

if(num > 0 && num <= Storage.size()) //условие допустимого товара.

{

CIN\_FLUSH;

cout << "Дата: "; getline(cin, x);

c.setDate(x);

cout << "товар: "; getline(cin, x);

c.setProduct(x);

cout << "Цена: "; getline(cin, x);

c.setPrice(x);

cout << "Количество: "; getline(cin, x);

c.setAmount(x);

Storage[num - 1] = c;

cout << endl << "Товар был изменен..." << endl << endl;

}

else

{

cout << "Введен некорректный номер товара..." << endl << endl;

}

}

else

{

cout << "На складе нет ни одного товара..." << endl << endl;

}

}

void deleteProduct() //функция удаляет избранный товар со склада.

{

system("cls");

int num;

if(Storage.size())

{

printProduct();

cout << "Введите номер товара, который хотите удалить: ";

cin >> num;

cout << endl;

if(num > 0 && num <= Storage.size())

{

auto first = Storage.cbegin(); // указатель на первый контакт.

Storage.erase(first + num - 1); // удаляется товар с номером num со склада.

cout << "Товар был удалён..." << endl << endl;

}

else

{

cout << "Введён некорректный номер товара..." << endl << endl;

}

}

else

{

cout << "На складе нет ни одного товара..." << endl << endl;

}

}

void deleteAllProduct() //функция полной очистки товаров со склада.

{

system("cls");

if(Storage.size())

{

Storage.clear(); //проводим чистку вектора.

cout << "Все товары были удалены со склада..." << endl << endl;

}

else

{

cout << "На складе нет ни одного товара..." << endl << endl;

}

}

void searchProductDate() //функция поиска товара со склада по дате.

{

system("cls");

string date; //переменная для ввода даты.

bool match = false; //переменная, показывающая нашлись ли нужные товары на складе.

int num; //переменная для нумерации найденных товаров на складе.

if(Storage.size())

{

CIN\_FLUSH;

cout << "Введите дату товара для поиска: ";

getline(cin, date);

cout << endl << "Совпадения:" << endl;

for (WareHouse c : Storage)

{

if (c.getDate().find(date) != string::npos)

{

match = true;

cout << (++num) << ". ";

cout << c.getDate() << " ";

cout << c.getProduct() << " ";

cout << c.getPrice() << " ";

cout << c.getAmount() << endl;

}

}

if(!match)

{

cout << "не найдены..." << endl;

}

cout << endl;

}

else

{

cout << "На складе нет ни одного товара..." << endl << endl;

}

}

void searchProducts() //функция поиска товара со склада по продукту.

{

system("cls");

string product;

bool match = false;

int num;

if(Storage.size())

{

CIN\_FLUSH;

cout << "Введите название товара для поиска: ";

getline(cin, product);

cout << endl << "Совпадения:" << endl;

for (WareHouse c : Storage)

{

if (c.getProduct().find(product) != string::npos)

{

match = true;

cout << (++num) << ". ";

cout << c.getDate() << " ";

cout << c.getProduct() << " ";

cout << c.getPrice() << " ";

cout << c.getAmount() << endl;

}

}

if(!match)

{

cout << "не найдены..." << endl;

}

cout << endl;

}

else

{

cout << "На складе нет ни одного товара..." << endl << endl;

}

}

void searchProductPrice() //функция поиска товара со склада по цене.

{

system("cls");

string price;

bool match = false;

int number;

if(Storage.size())

{

CIN\_FLUSH; // очистка потока ввода

cout << "Введите название товара для поиска: ";

getline(cin, price);

cout << endl << "Совпадения:" << endl;

for (WareHouse c : Storage)

{

if (c.getPrice().find(price) != string::npos)

{

match = true;

cout << (++number) << ". ";

cout << c.getDate() << " ";

cout << c.getProduct() << " ";

cout << c.getPrice() << " ";

cout << c.getAmount() << endl;

}

}

if(!match)

{

cout << "не найдены..." << endl;

}

cout << endl;

}

else

{

cout << "На складе нет ни одного товара..." << endl << endl;

}

}

void searchProductAmount() //функция поиска товара со склада по его количеству.

{

system("cls");

string amount;

bool match = false;

int number;

if(Storage.size())

{

CIN\_FLUSH;

cout << "Введите название товара для поиска: ";

getline(cin, amount);

cout << endl << "Совпадения:" << endl;

for (WareHouse c : Storage)

{

if (c.getAmount().find(amount) != string::npos)

{

match = true;

cout << (++number) << ". ";

cout << c.getDate() << " ";

cout << c.getProduct() << " ";

cout << c.getPrice() << " ";

cout << c.getAmount() << endl;

}

}

if(!match)

{

cout << "не найдены..." << endl;

}

cout << endl;

}

else

{

cout << "На складе нет ни одного товара..." << endl << endl;

}

}

void getProductFromFile() //функция для записи данных из файла.

{

system("cls");

Storage.clear();

ifstream fin("Storage.txt"); //открываем файл списка товаров со склада для чтения.

string line; //строка для считывания контактов из файла.

WareHouse c; //объект для записи в него товара со склада.

string x; //переменная для получения данных товара из тектстового потока.

if(!fin.is\_open())

{

cerr << "ОШИБКА: не удалось открыть файл для получения информации о товарах..." << endl;

exit(EXIT\_FAILURE); //завершение программы с ошибкой.

}

else

{

while(getline(fin, line)) //читаем по одному товару.

{

stringstream ss(line); //делаем из строки строковый поток.

ss >> x; c.setDate(x); //переносим из строкового потока данные в объект.

ss >> x; c.setProduct(x);

ss >> x; c.setPrice(x);

ss >> x; c.setAmount(x);

Storage.push\_back(c); //записываем объект в складское приложение.

}

fin.close(); //закрытие файла.

cout << "Товары были загружены из файла..." << endl << endl;

}

}

void writeProductInFile() //функция для записи данных в файл.

{

system("cls");

ofstream fout("Storage.txt"); //открываем файл списка товаров со склада на запись.

string product; //переменная для записи товара со склада.

if(!fout.is\_open())

{

cerr << "ОШИБКА: не удалось открыть файл для сохранения информации о товарах..." << endl;

exit(EXIT\_FAILURE); //завершение программы с ошибкой.

}

else

{

for(WareHouse c : Storage) //проходим по всем контактам из складского приложения.

{

product = c.getDate(); //формируем строку товара со склада.

product += " ";

product += c.getProduct();

product += " ";

product += c.getPrice();

product += " ";

product += c.getAmount();

fout << product << endl; //записываем строку в файл.

}

fout.close(); //закрытие файла.

cout << "Товары были сохранены в файл..." << endl << endl;

}

}

//Далее идёт аналогичное описание, которое можно опустить.

void addProductSale()

{

system("cls");

string x;

Products c;

CIN\_FLUSH;

cout << "Продаваемая продукция..." << endl << endl;

cout << "Дата: "; getline(cin, x);

c.setDate(x);

cout << "товар: "; getline(cin, x);

c.setProduct(x);

cout << "Цена: "; getline(cin, x);

c.setPrice(x);

cout << "Фирма: "; getline(cin, x);

c.setFirm(x);

cout << "Количество: "; getline(cin, x);

c.setSale(x);

Sales.push\_back(c);

cout << endl << "Продукция успешно продана..." << endl << endl;

}

void printProductSale()

{

system("cls");

int num = 0;

if(Sales.size())

{

cout << "=========================================================================================" << endl;

cout << "= №\tДата\t\tПродукт\t\tЦена\t\tФирма\t\tКоличество\t=" << endl;

cout << "=========================================================================================" << endl;

for (Products c : Sales)

{

cout << "= " << (++num) << ".\t";

cout << c.getDate() << "\t";

if(c.getDate().length() < 8) cout << "\t";

cout << c.getProduct() << "\t";

if(c.getProduct().length() < 8) cout << "\t";

cout << c.getPrice() << "\t";

if(c.getPrice().length() < 8) cout << "\t";

cout << c.getFirm() << "\t";

if(c.getFirm().length() < 8) cout << "\t";

cout << c.getSale() << "\t";

if(c.getSale().length() < 8) cout << "\t";

cout << "=" << endl;

}

cout << "=========================================================================================" << endl << endl;

}

else

{

cout << "Ничего не продано..." << endl;

}

cout << endl;

}

void fixProductSale()

{

system("cls");

int num;

Products c;

string attr;

if(Sales.size())

{

printProductSale();

cout << "Введите номер проданного товара, который хотите отредактировать: ";

cin >> num;

cout << endl;

if(num > 0 && num <= Sales.size())

{

CIN\_FLUSH;

cout << "Дата: "; getline(cin, attr);

c.setDate(attr);

cout << "товар: "; getline(cin, attr);

c.setProduct(attr);

cout << "Цена: "; getline(cin, attr);

c.setPrice(attr);

cout << "Фирма: "; getline(cin, attr);

c.setFirm(attr);

cout <<"Количество: "; getline(cin, attr);

c.setSale(attr);

Sales[num - 1] = c;

cout << endl << "Проданный товар был изменен..." << endl << endl;

}

else

{

cout << "Введен некорректный номер проданного товара..." << endl << endl;

}

}

else

{

cout << "Ничего не продано..." << endl << endl;

}

}

void deleteProductSale()

{

system("cls");

int num;

if(Sales.size())

{

printProductSale();

cout << "Введите номер проданного товара, который хотите удалить: ";

cin >> num;

cout << endl;

if(num > 0 && num <= Sales.size())

{

auto first = Sales.cbegin();

Sales.erase(first + num - 1);

cout << "Проданный товар был удалён..." << endl << endl;

}

else

{

cout << "Введён некорректный номер проданного товара..." << endl << endl;

}

}

else

{

cout << "Ничего не продано..." << endl << endl;

}

}

void deleteAllProductSale()

{

system("cls");

if(Sales.size())

{

Sales.clear();

cout << "Все проданные товары были удалены из списка..." << endl << endl;

} else

{

cout << "Ничего не продано..." << endl << endl;

}

}

void searchProductSaleDate()

{

system("cls");

string date;

bool match = false;

int num;

if(Sales.size())

{

CIN\_FLUSH;

cout << "Введите дату проданного товара для поиска: ";

getline(cin, date);

cout << endl << "Совпадения:" << endl;

for (Products c : Sales)

{

if (c.getDate().find(date) != string::npos)

{

match = true;

cout << (++num) << ". ";

cout << c.getDate() << " ";

cout << c.getProduct() << " ";

cout << c.getPrice() << " ";

cout << c.getFirm() << " ";

cout << c.getSale() << endl;

}

}

if(!match)

{

cout << "не найдены..." << endl;

}

cout << endl;

}

else

{

cout << "Ничего не продано..." << endl << endl;

}

}

void searchProductsSale()

{

system("cls");

string product;

bool match = false;

int num;

if(Sales.size())

{

CIN\_FLUSH;

cout << "Введите дату проданного товара для поиска: ";

getline(cin, product);

cout << endl << "Совпадения:" << endl;

for (Products c : Sales)

{

if (c.getProduct().find(product) != string::npos)

{

match = true;

cout << (++num) << ". ";

cout << c.getDate() << " ";

cout << c.getProduct() << " ";

cout << c.getPrice() << " ";

cout << c.getFirm() << " ";

cout << c.getSale() << endl;

}

}

if(!match)

{

cout << "не найдены..." << endl;

}

cout << endl;

}

else

{

cout << "Ничего не продано..." << endl << endl;

}

}

void searchProductSalePrice()

{

system("cls");

string price;

bool match = false;

int num;

if(Sales.size())

{

CIN\_FLUSH;

cout << "Введите дату проданного товара для поиска: ";

getline(cin, price);

cout << endl << "Совпадения:" << endl;

for (Products c : Sales)

{

if (c.getPrice().find(price) != string::npos)

{

match = true;

cout << (++num) << ". ";

cout << c.getDate() << " ";

cout << c.getProduct() << " ";

cout << c.getPrice() << " ";

cout << c.getFirm() << " ";

cout << c.getSale() << endl;

}

}

if(!match)

{

cout << "не найдены..." << endl;

}

cout << endl;

}

else

{

cout << "Ничего не продано..." << endl << endl;

}

}

void searchProductSaleFirm()

{

system("cls");

string firm;

bool match = false;

int num;

if(Sales.size())

{

CIN\_FLUSH;

cout << "Введите дату проданного товара для поиска: ";

getline(cin, firm);

cout << endl << "Совпадения:" << endl;

for (Products c : Sales)

{

if (c.getFirm().find(firm) != string::npos)

{

match = true;

cout << (++num) << ". ";

cout << c.getDate() << " ";

cout << c.getProduct() << " ";

cout << c.getPrice() << " ";

cout << c.getFirm() << " ";

cout << c.getSale() << endl;

}

}

if(!match)

{

cout << "не найдены..." << endl;

}

cout << endl;

}

else

{

cout << "Ничего не продано..." << endl << endl;

}

}

void searchProductSales()

{

system("cls");

string sale;

bool match = false;

int num;

if(Sales.size())

{

CIN\_FLUSH;

cout << "Введите дату проданного товара для поиска: ";

getline(cin, sale);

cout << endl << "Совпадения:" << endl;

for (Products c : Sales)

{

if (c.getSale().find(sale) != string::npos)

{

match = true;

cout << (++num) << ". ";

cout << c.getDate() << " ";

cout << c.getProduct() << " ";

cout << c.getPrice() << " ";

cout << c.getFirm() << " ";

cout << c.getSale() << endl;

}

}

if(!match)

{

cout << "не найдены..." << endl;

}

cout << endl;

}

else

{

cout << "Ничего не продано..." << endl << endl;

}

}

void getProductSaleFromFile()

{

system("cls");

Sales.clear();

ifstream fin("Sales.txt");

string line;

Products c;

string attr;

if(!fin.is\_open())

{

cerr << "ОШИБКА: не удалось открыть файл для получения информации о товарах..." << endl;

exit(EXIT\_FAILURE);

} else

{

while(getline(fin, line))

{

stringstream ss(line);

ss >> attr; c.setDate(attr);

ss >> attr; c.setProduct(attr);

ss >> attr; c.setPrice(attr);

ss >> attr; c.setFirm(attr);

ss >> attr; c.setSale(attr);

Sales.push\_back(c);

}

fin.close();

cout << "Проданные товары были загружены из файла..." << endl << endl;

}

}

void writeProductSaleInFile()

{

system("cls");

ofstream fout("Sales.txt");

string product;

if(!fout.is\_open())

{

cerr << "ОШИБКА: не удалось открыть файл для сохранения информации о товарах..." << endl;

exit(EXIT\_FAILURE);

}

else

{

for(Products c : Sales)

{

product = c.getDate();

product += " ";

product += c.getProduct();

product += " ";

product += c.getPrice();

product += " ";

product += c.getFirm();

product += " ";

product += c.getSale();

fout << product << endl;

}

fout.close();

cout << "Проданные товары были сохранены в файл..." << endl << endl;

}

}

int main ()

{

system("mode con cols=89 lines=50"); //установка размера консоли.

system("color 1F"); //установка цвета консоли и шрифта.

setlocale(LC\_ALL,"rus"); //функция для использования русских букв.

SetConsoleCP(1251); //установка кодировок для ввода/вывода консоли.

SetConsoleOutputCP(1251);

int x; //переменная для выбора пунктов меню.

do { //цикл с постусловием.

menu(); //выводим меню.

cin >> x; //пользователь выбирает пункт меню.

switch(x)

{

case 1:

addProduct();

break;

case 2:

printProduct();

break;

case 3:

fixProduct();

break;

case 4:

deleteProduct();

break;

case 5:

deleteAllProduct();

break;

case 6:

searchProductDate();

break;

case 7:

searchProducts();

break;

case 8:

searchProductPrice();

break;

case 9:

searchProductAmount();

break;

case 10:

getProductFromFile();

break;

case 11:

writeProductInFile();

break;

case 12:

addProductSale();

break;

case 13:

printProductSale();

break;

case 14:

fixProductSale();

break;

case 15:

deleteProductSale();

break;

case 16:

deleteAllProductSale();

break;

case 17:

searchProductSaleDate();

break;

case 18:

searchProductsSale();

break;

case 19:

searchProductSalePrice();

break;

case 20:

searchProductSaleFirm();

break;

case 21:

searchProductSales();

break;

case 22:

getProductSaleFromFile();

break;

case 23:

writeProductSaleInFile();

break;

case 0:

{

system("cls"); //очистка консоли.

cout << "Завершение работы программы..." << endl;

cin.get();

return EXIT\_SUCCESS; //успешное завершение главной функции.

}

}

} while(true);

return 0;

}

**Код реализуемых классов**

**1.**

#ifndef GOODSRECEIPT\_H\_INCLUDED //директива препроцессора: осуществляет условную компиляцию при неопределённости символической константы.

#define GOODSRECEIPT\_H\_INCLUDED //директива препроцессора: задаёт макроопределение (макрос) или символическую константу.

using namespace std; //пространство стандартных имён.

#include<iostream> //заголовочный файл с классами, функциями и переменными для организации ввода-вывода в языке программирования C++.

class GoodsReceipt //объявления класса GoodsReceipt.

{

protected: //модификатор доступа protected: доступ открыт классам, производным от данного.

string Date; //объявление поля класса.

public: //модификатор доступа public: доступ открыт всем, кто видит определение данного класса.

GoodsReceipt() //конструктор без параметров.

{

setDate("");

}

GoodsReceipt(string receipt\_date) //конструктор с параметрами.

{

setDate(receipt\_date);

}

void setDate(string receipt\_date) //метод setDate.

{

this->Date = receipt\_date;

}

string getDate() const //метод getDate.

{

return Date;

}

GoodsReceipt(const GoodsReceipt &pp) //конструктор копирования.

{

this->Date = pp.Date;

}

~GoodsReceipt() //деструктор без реализации.

{

}

};

#endif // GOODSRECEIPT\_H\_INCLUDED //директива препроцессора: конец ветки условной компиляции.

//Описывать все классы не будем, потому что они однотипны друг другу.

**2.**

#ifndef SALESGOODS\_H\_INCLUDED

#define SALESGOODS\_H\_INCLUDED

using namespace std;

#include<iostream>

#include"GoodsReceipt.h"

class SalesGoods : public GoodsReceipt //объявления класса SalesGoods, наследуемый класс GoodsReceipt.

{

protected:

string Product;

string Price;

public:

SalesGoods() : GoodsReceipt("")

{

setProduct("");

setPrice("");

}

SalesGoods(string receipt\_date, string product\_received, string price\_product) : GoodsReceipt(receipt\_date)

{

setProduct(product\_received);

setPrice(price\_product);

}

void setProduct(string product\_received)

{

this->Product = product\_received;

}

void setPrice(string price\_product)

{

this->Price = price\_product;

}

string getProduct() const

{

return Product;

}

string getPrice() const

{

return Price;

}

SalesGoods(const SalesGoods &pp)

{

this->Date = pp.Date;

this->Product = pp.Product;

this->Price = pp.Price;

}

~SalesGoods()

{

}

};

#endif // SALESGOODS\_H\_INCLUDED

**3.**

#ifndef WAREHOUSE\_H\_INCLUDED

#define WAREHOUSE\_H\_INCLUDED

using namespace std;

#include<iostream>

#include"SalesGoods.h"

class WareHouse : public SalesGoods

{

protected:

string Amount;

public:

WareHouse() : SalesGoods("","","")

{

setAmount("");

}

WareHouse(string receipt\_date, string product\_received, string price\_product, string quantity\_of\_goods) : SalesGoods(receipt\_date, product\_received, price\_product)

{

setAmount(quantity\_of\_goods);

}

void setAmount(string quantity\_of\_goods)

{

this->Amount = quantity\_of\_goods;

}

string getAmount() const

{

return Amount;

}

WareHouse(const WareHouse &pp)

{

this->Date = pp.Date;

this->Product = pp.Product;

this->Price = pp.Price;

this->Amount = pp.Amount;

}

~WareHouse()

{

}

};

#endif // WAREHOUSE\_H\_INCLUDED

**4.**

#ifndef COUNTERPARTY\_H\_INCLUDED

#define COUNTERPARTY\_H\_INCLUDED

using namespace std;

#include<iostream>

#include"SalesGoods.h"

class CounterParty : public SalesGoods

{

protected:

string Firm;

public:

CounterParty() : SalesGoods("","","")

{

setFirm("");

}

CounterParty(string receipt\_date, string product\_received, string price\_product, string company\_name) : SalesGoods(receipt\_date, product\_received, price\_product)

{

setFirm(company\_name);

}

void setFirm(string company\_name)

{

this->Firm = company\_name;

}

string getFirm() const

{

return Firm;

}

CounterParty(const CounterParty &pp)

{

this->Date = pp.Date;

this->Product = pp.Product;

this->Price = pp.Price;

this->Firm = pp.Firm;

}

~CounterParty()

{

}

};

#endif // COUNTERPARTY\_H\_INCLUDED

**5.**

#ifndef PRODUCTS\_H\_INCLUDED

#define PRODUCTS\_H\_INCLUDED

using namespace std;

#include<iostream>

#include"CounterParty.h"

class Products : public CounterParty

{

private: //модификатор доступа private: доступ открыт самому классу (т.е. функциям-членам данного класса) и друзьям (friend) данного класса - как функциям, так и классам.

string Sale;

public:

Products() : CounterParty("","","","")

{

setSale("");

}

Products(string receipt\_date, string product\_received, string price\_product, string company\_name, string sale\_of\_goods) : CounterParty(receipt\_date, product\_received, price\_product, company\_name)

{

setSale(sale\_of\_goods);

}

void setSale(string sale\_of\_goods)

{

this->Sale = sale\_of\_goods;

}

string getSale() const

{

return Sale;

}

Products(const Products &pp)

{

this->Date = pp.Date;

this->Product = pp.Product;

this->Price = pp.Price;

this->Firm = pp.Firm;

this->Sale = pp.Sale;

}

~Products()

{

}

};

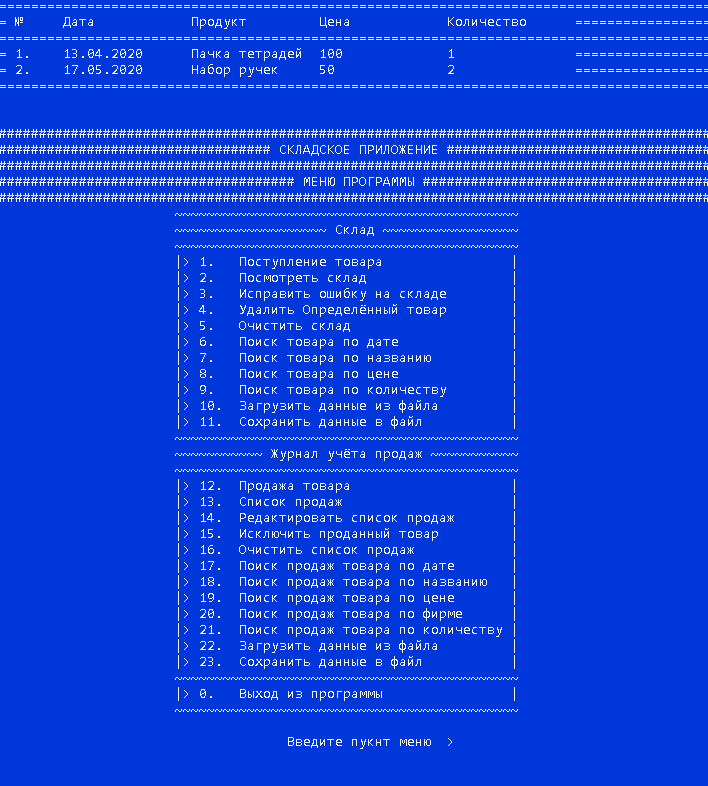
#endif // PRODUCTS\_H\_INCLUDED

**Меню программы**



**Пример работы с программой**

Загрузка данных, просмотр и изменение товара со склада.



**Список используемой литературы (ссылок)**

1. <https://www.cyberforum.ru/>
2. <https://code-live.ru/tag/cpp-manual/>
3. <http://cppstudio.com/cat/274/>
4. <https://prog-cpp.ru/cpp/>