**Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 13.03.02 – Электроэнергетика и электротехника | |
| **Профиль** | Электропривод и автоматика | |
| **Факультет** | ФЭА | |
| **Кафедра** | РАПС | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | Белов М.П. |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРА**

**Тема: Разработка системы управления складированием продукции**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | Ведерников Б. |
|  |  | *подпись* |  |  |
| Руководитель | к.т.н. доцент |  |  | Шевченко А.В. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
| Консультанты | к.т.н. доцент |  |  | Козлова Л.П. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
|  | ассистент |  |  | Подлевских М.Г. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |

Санкт-Петербург

2022

**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой РАПС |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Белов М.П. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | Ведерников Б. | | | |  | Группа | 8403 |
| Тема работы: Разработка системы управления складированием продукции | | | | | | | |
| Место выполнения ВКР: СПбГЭТУ «ЛЭТИ» | | | | | | | |
| Исходные данные (технические требования):  Разработать систему управления складированием продукции, оценить затраты на ее разработку | | | | | | | |
| Содержание ВКР:  Основные принципы складирования, создание виртуальной модели складского помещения, описание важнейших этапов разработки, результаты разработки. | | | | | | | |
| Перечень отчетных материалов: текст ВКР, иллюстративный материал, отзыв научного руководителя, пояснительная записка. | | | | | | | |
| Дополнительные разделы: Технико-экономическое обоснование работы | | | | | | | |
|  | | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | | |
| «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. | | | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. | | | | |
|  | | |  | | | | |
| Студент | |  | | Ведерников Б. | | | |
| Руководитель к.т.н. доцент | |  | | Шевченко А.В. | | | |
| *(Уч. степень, уч. звание)* | |  | |  | | | |

**календарный план выполнения**

**выпускной квалификационной работы**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой РАПС |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Белов М.П. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Ведерников Б. |  | Группа | 8403 |
| Тема работы: Разработка системы управления складированием продукции | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Обзор литературы по теме работы | 10.01 – 10.02 |
| 2 | Анализ и структуризация ВКР | 20.02 – 20.03 |
| 3 | Написание рабочей программы | 25.03 – 05.05 |
| 4 | Написание основного и дополнительного раздела ВКР | 10.05 – 15.05 |
| 5 | Оформление пояснительной записки | 28.05 – 29.05 |
| 6 | Оформление иллюстративного материала | 30.05 – 31.05 |

|  |  |  |
| --- | --- | --- |
| Студент |  | Ведерников Б. |
| Руководитель к.т.н., доцент |  | Шевченко А.В. |

*(Уч. степень, уч. звание)*

**РЕФЕРАТ**

Пояснительная записка 62 стр., 29 рис., 11 табл., 9 ист., 9 прил.

ТАБЛИЦА, модель, База данных, программа, АЛГОРИТМ

Объектом разработки является система управления складированием продукции.

Цель работы – разработка бесплатной программы для обеспечения удобного функционирования складских помещений с понятным интерфейсом, создание базы данных для учета продукции, реализация алгоритма генерации QR-кодов для маркировки товаров.

Работа включает в себя общее описание работы складов, способов складирования продукции, а также создание виртуальной модели складского помещения на уровне процессов. Была продемонстрирована декомпозиция каждого из представленных в модели процесса.

Для разработки программы использовалась среда разработки приложений Embarcadero C++ Builder, компоненты для работы с базами данных ADO и библиотека libqrencode для создания генерации QR-кодов.

Итогом разработки стала полностью функционирующая система управления складированием продукции с удобным и интуитивно понятным интерфейсом, которая работает с базами данных, помогая вести учет продукции на складе, проводить анализ закупок на основе графической информации, и генерировать QR-коды для маркировки товаров.

**ABSTRACT**

Since the introduction of computer technologies into production, there has been a demand for the development of various kinds of software specifically for enterprises. The widespread use of home computers and the ability to provide them to almost everyone has significantly increased the number of people able to engage in such development. The development of technologies has brought the quality of developed applications to a new level and led to the emergence of so-called rapid application development tools (in particular, the Embarcadero C++ Builder used in work), which further reduce the entry threshold, simplifying the process of program development.

In parallel with the development of digital technologies, the country has experienced several crises, the last of which especially showed a shortage of inexpensive and simple domestic developments, including for small enterprises, which are particularly affected by any economic shocks. Unfortunately, most of the similar programs developed in the WRC are paid, or have very complex functionality that is redundant for small workshops. Among other things, almost every enterprise needs warehouses regardless of the products produced, which proves the universality of the created product warehousing management system (PWM).

It is equally important to note that the company's profit directly depends on the efficiency of the organization of storage facilities. The ability to save money on the purchase of a program license is also a significant plus.

For this task, as mentioned earlier, the Embarcadero C++ Builder environment was chosen. It is very convenient and relatively easy to learn, has at its disposal sufficient functionality for work. The bonus to everything was that the program can be used temporarily for free, which reduced the budget for the development of the PWM.

**содержание**

[ВВЕДЕНИЕ 8](#_Toc104479907)

[1 НЕОБХОДИМАЯ ИНФОРМАЦИЯ О СКЛАДИРОВАНИИ ПРОДУКЦИИ 9](#_Toc104479908)

[1.1 Основные принципы складирования 9](#_Toc104479909)

[1.2 Создание виртуальной модели складского помещения 12](#_Toc104479910)

[2 ОПИСАНИЕ ВАЖНЕЙШИХ ЭТАПОВ РАЗРАБОТКИ 20](#_Toc104479911)

[2.1 Обзор и выбор технологии доступа к данным из БД 20](#_Toc104479912)

[2.2 Описание работы алгоритма генерации QR-кодов 24](#_Toc104479913)

[3 РЕЗУЛЬТАТЫ РАЗРАБОТКИ 36](#_Toc104479914)

[3.1 Описание работы программы 36](#_Toc104479915)

[3.2 Тестирование СУСП 44](#_Toc104479916)

[4 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ВКР 47](#_Toc104479917)

[4.1 Введение 47](#_Toc104479918)

[4.2 Составление детализированного плана работ 47](#_Toc104479919)

[4.3 Основная заработная плата 48](#_Toc104479920)

[4.4 Дополнительная заработная плата 49](#_Toc104479921)

[4.5 Отчисления на социальные нужды 50](#_Toc104479922)

[4.6 Сырье и материалы 50](#_Toc104479923)

[4.7 Расчет затрат на эксплуатацию оборудования и его содержание 51](#_Toc104479924)

[4.8 Расчёт амортизационных отчислений 52](#_Toc104479925)

[4.9 Калькуляция затрат на ВКР 53](#_Toc104479926)

[ЗАКЛЮЧЕНИЕ 54](#_Toc104479927)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 55](#_Toc104479928)

[ПРИЛОЖЕНИЕ А 56](#_Toc104479929)

# ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей пояснительной записке применяют следующие термины с соответствующими определениями:

ADO – ActiveX Data Objects

BDE - Borland Database Engine

COM – Component Object Model

DAO - Data Access Objects

ISAM – Indexed Sequential Access Method

MDAC – Microsoft Data Access Components

OLE DB – Object Linking and Embedding DataBase

ODBC – Open DataBase Connectivity

QR - Quick Response

RDO - Remote Data Objects

VCL - Visual Component Library

БД – База Данных

ОС – Операционная система

ПО – Программное Обеспечение

СУБД – Система Управления Базами Данных

СУСП – Система Управления Складированием Продукции

# ВВЕДЕНИЕ

С момента внедрения компьютерных технологий в производство, появился спрос на разработку различного рода ПО специально для предприятий. Широкое распространение домашних компьютеров и возможность получить их практически каждому в разы увеличило количество людей, способных заниматься такой разработкой. Развитие технологий вывело качество разрабатываемых приложений на новый уровень и привело к появлению так называемых инструментов быстрой разработки приложений (в частности используемый в работе Embarcadero C++ Builder), которые еще больше понижают порог вхождения, упрощая процесс разработки программ.

Параллельно с развитием цифровых технологий страна пережила несколько кризисов, последний из которых особенно показал нехватку отечественных недорогих и простых разработок, в том числе и для маленьких предприятий, которые особенно страдают от любых экономических потрясений. К сожалению, большинство аналогичных разработанной в ВКР программ являются платными, либо имеют очень сложный функционал, который является избыточным для маленьких цехов. Помимо прочего, в складах нуждается практически каждое предприятие независимо от производимой продукции, что доказывает универсальность созданной СУСП.

Не менее важно отметить, что от эффективности организации складских помещений зависит напрямую прибыль предприятия. Возможность сэкономить на покупке лицензии программы также является весомым плюсом.

Для поставленной задачи, как уже было сказано ранее, была выбрана среда Embarcadero C++ Builder. Она очень удобна и относительно проста в освоении, имеет в своем распоряжении достаточный для работы функционал. Бонусом ко всему стало, то, что программой можно пользоваться временно бесплатно, что сократило бюджет разработки СУСП.

# 1 НЕОБХОДИМАЯ ИНФОРМАЦИЯ О СКЛАДИРОВАНИИ ПРОДУКЦИИ

## 1.1 Основные принципы складирования

Упорядоченное ведение складского хозяйства – один из основных принципов эффективного управления ресурсами склада. Создать порядок – значит создать мотивацию работников к аккуратному обращению с продукцией, упорядочить процессы хранения запасов и введения новых товаров в оборот, разместить продукцию по приоритетности. Можно по-разному достичь этих целей, но главное – результат, то есть порядок. Как правило, появление порядка на складе даёт прямой экономический эффект в виде экономии места в складском помещении, росту оборачиваемости товаров, увеличению прибыли.

Складское хозяйство предприятия является звеном, которому надо уделять особое внимание. Доказательством важности этого звена является схема, указанная на рисунке 1 и показывающая “круговорот” финансовых и материальных потоков в одном предприятии.

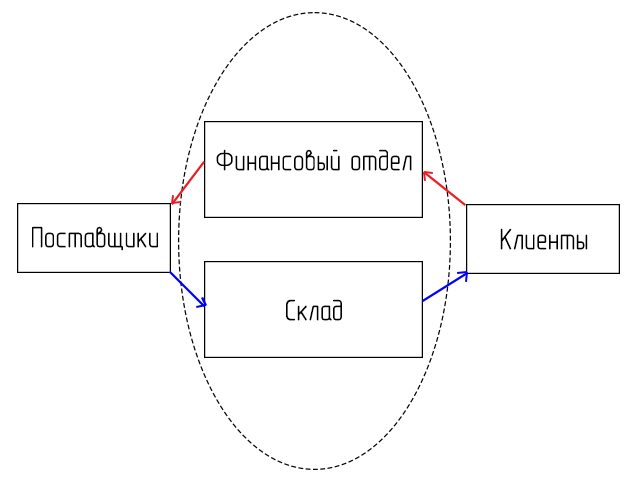


Рисунок 1 – Схема оборота финансовых и материальных потоков предприятия

Красные стрелочки означают финансовые потоки, а синие — материальные потоки. То, что в качестве финансового потока уходит к поставщикам, возвращается на предприятия в качестве материальных ценностей (товаров) и поступает на склад. С другой стороны, все, что уходит к покупателям, возвращается на предприятие в виде финансового потока.

Безусловно, схема условная, так как она не отражает, последовательность потоков, в ней нет коммерческого отдела, без которого немыслим процесс. И тем не менее схема наглядно показывает роль склада.

Финансовые потоки любого предприятия регламентированы законами, а материальные потоки в основном — внутренними процессами. Точкой соприкосновения двух основных типов материальных потоков — входящих и выходящих — является склад. Иными словами, склад – это звено, в котором сконцентрированы процедуры, касающиеся не только самого склада, но и его взаимодействия с остальными звеньями предприятия [1].

В цепочке дистрибуции товаров затраты на складское хранение могут достигать 50% их стоимости. Поэтому сведение к минимуму этих издержек является важным конкурентным фактором для любого предприятия. Помочь в сокращении расходов может информация о том, какие складские системы хранения товаров наиболее приемлемы для конкретного ассортимента продукции. Правильный выбор стеллажей может на порядок сократить затраты на персонал и увеличить эффективную площадь склада.

При организации функционирования склада необходимо сразу продумать, каким образом будет храниться продукция.

Основными видами складских систем хранения являются:

1. Стеллажи
2. Напольное хранение
3. Вешала
4. Ёмкости для жидкостей
5. Крюки и другие системы для специфических видов продукции

В разработанной системе управления складированием продукции в качестве примера товарами являются металлы, систему хранения которых и следует рассмотреть [2].

В зависимости от вида, марки металла, размеров проката, его упаковки, способов транспортирования и др. должно производиться в штабелях (рисунок 2) или на стеллажах, в таре, в связках или единичными грузами на открытых площадках, закрытых и полузакрытых складах.



Рисунок 2 – Арматурная сетка в штабеле

Штабелирование является допустимой складской системой хранения для помещений небольшой высоты. Оно может быть трёх видов:

1. Прямое
2. Перекрестное
3. Обратное

Выбор складской системы хранения на полу производится, учитывая обеспечение максимально возможной устойчивости штабеля. Минусом такого способа укладки является большой вес, который оказывает давление на нижние ярусы товаров, что ограничивает максимальную высоту штабеля.

Вдобавок, некоторые товары требуют регулярного проветривания, из-за чего их нельзя складывать на полу в большом объеме в одном месте.

Таким образом, при острой необходимости дополнительных площадей или дорогой арендной плате предприятие всегда может оптимизировать свои расходы, инвестируя в более совершенные складские системы хранения товаров. Главное – рассчитать затраты на окупаемость проведения модернизации. Так как разрабатываемая система управления складированием продукции нацелена скорее на небольшие предприятия, то с большой вероятностью именно такой тип хранения материалов будет использован на производстве.

## 1.2 Создание виртуальной модели складского помещения

Складирование — это логистическая операция, заключающаяся в содержании запасов участниками логистического канала и обеспечивающая сохранность запасов, их рациональное размещение, учет, постоянное обновление и безопасные методы работы.

У каждого склада имеется целый ряд функций:

1. складирование и хранение товаров;
2. контроль над поставками и формирование ассортимента.
3. унитизация грузов;
4. приём и отгрузка товаров
5. другие логистические услуги, в том числе, тестирование работы приборов, сборка или фасовка продукции.

Для более наглядного понимания совокупности процессов складирования, была разработана виртуальная модель складского помещения. Непосредственно создание модели происходило в программе “Система планирования ресурсов и сопровождения заказов”, разработанной на кафедре РАПС и представляющей собой среду развертывания приложений для разработки приложений интегрированных систем управления. Эта среда содержит встроенную базу данных, содержащую информацию о конфигурации системы (составе технических средств, приложений, пользователей, их прав доступа и привилегий). Непосредственное создание модели процесса складирования производилось во встроенной программе «Процессы». Полученная схема представлена на рисунке 3.

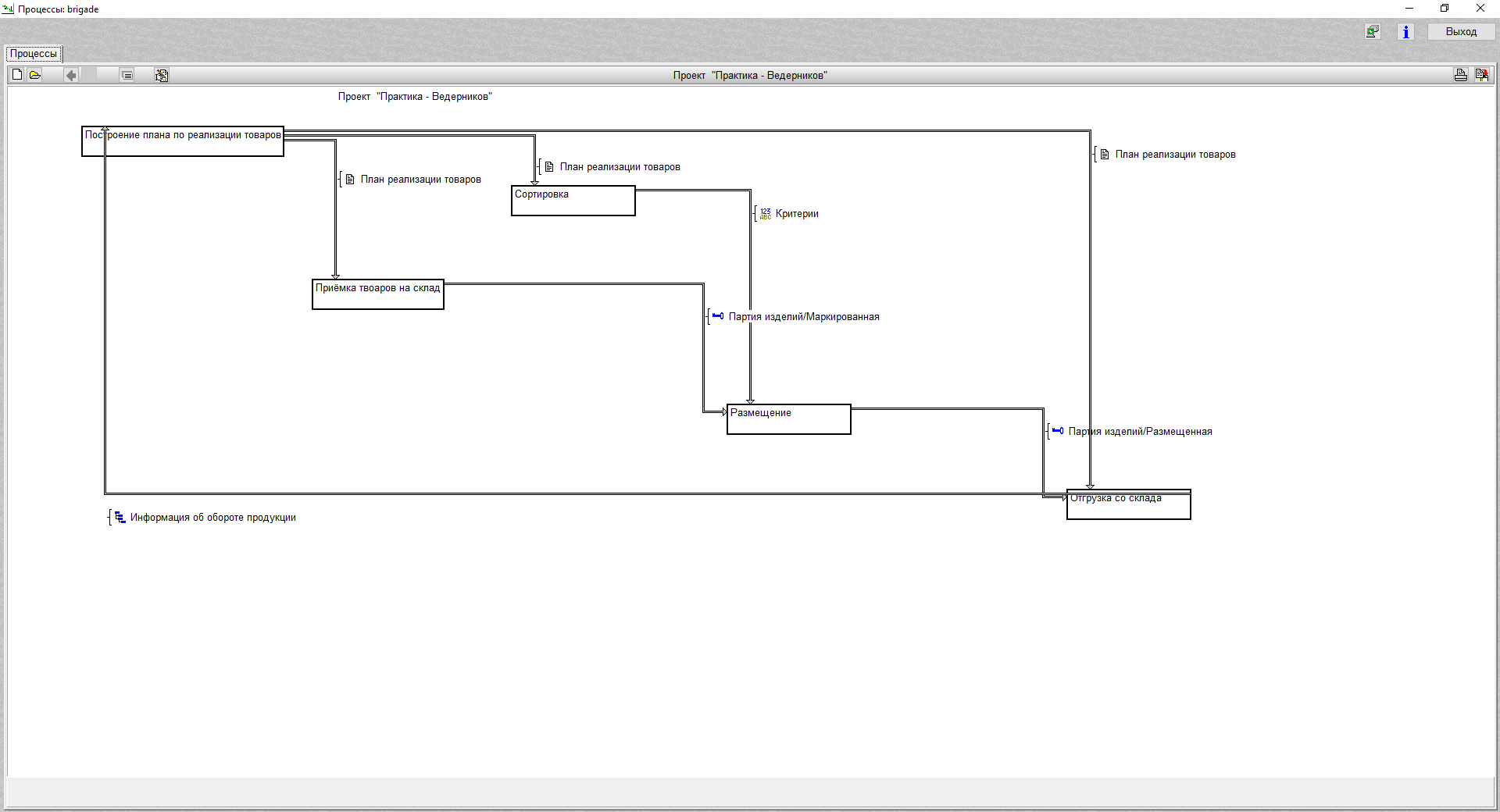


Рисунок 3 – Модель процесса складирования

На приведенной схеме имеется 5 процессов и 4 различных объекта. Процессы: построение плана по реализации товаров, приёмка товаров на склад, сортировка, размещение, отгрузка со склада. Объекты: план реализации товаров, критерии, партия изделий и информация об обороте продукции. Информационный объект «План реализации товаров» является выходом процесса «Построение плана по реализации товаров» и управляет процессами «Приёмка товаров на склад», «Сортировка» и «Отгрузка со склада». Информационный объект «Критерии» является выходом процесса «Сортировка» и управляет процессом «Размещение». Партия изделий является материальным объектом и имеет 3 фазы жизненного цикла: полученная, маркированная и размещенная. Объект «Информация об обороте продукции» является выходом процесса «Отгрузка со склада» и управляет процессом «Построение плана по реализации товаров».

Программа позволяет выполнить декомпозицию любого процесса на нескольких уровнях, при этом число уровней условно бесконечно и ограничивается принципом разумной достаточности, чтобы детальность описания не перегрузила модель и не помешала ее целостному восприятию [3]. Благодаря такой возможности были разработаны окна подпроцессов каждого процесса, представленного на рисунке 3.

1) Единственный процесс, который не имеет внутри себя подпроцессов – построение плана по реализации товаров, поэтому его окно выглядит соответствующим образом (рисунок 4). На основе информации, полученной с выхода процесса «Отгрузка со склада», формируется план по реализации товаров. Например, предприятие начнет производить больше таких товаров, доля продаж на которые была наибольшей, и меньше производить продукции, приносящей меньше прибыли.

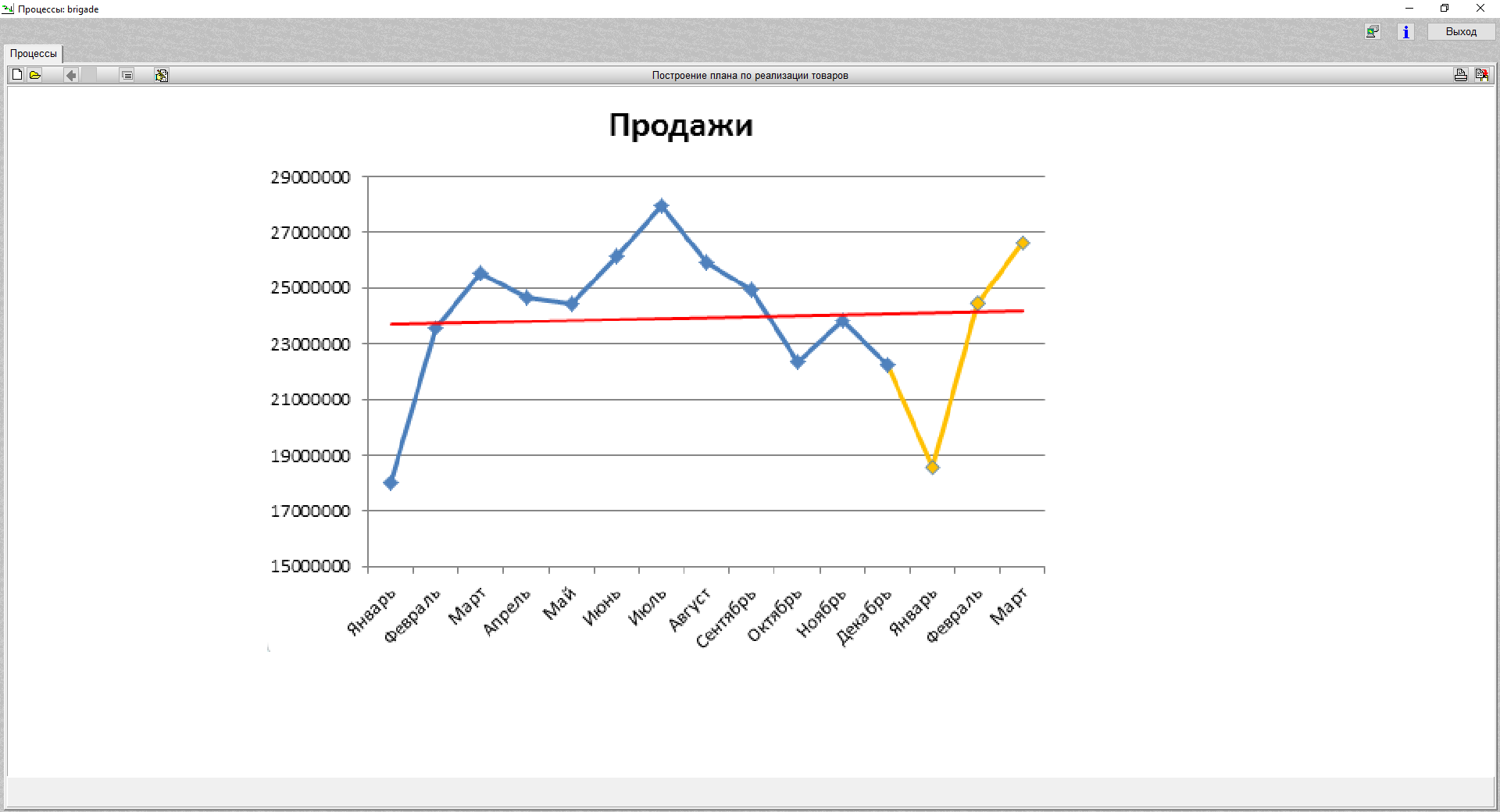


Рисунок 4 – Процесс «Построение плана по реализации товаров»

2) На уровне процесса «Приёмка товаров на склад» находятся 3 подпроцесса, а также заметен переход фаз жизненного цикла объекта «Партия изделий» (рисунок 5). Информационный объект «План по реализации товаров» управляет процессами «Приёмка товара по качеству и количеству» и «Документальное оформление результатов приёмки». Материальный объект «Партия изделий» при этом создается здесь на выходе процесса «Разгрузка транспорта» и является выходом в фазе «Маркированная» подпроцесса «Документальное оформление результатов приёмки» процесса «Приёмка товаров на склад».

Справа в углу реализован элемент интерфейса основной программы. Здесь при получении партии продукции на склад, работник вводит соответствующие товару характеристики, которые заносятся в базу данных продукции склада, на основании которых программа генерирует уникальный QR-код, используемый для маркировки каждого отдельного товара (рисунок 6).

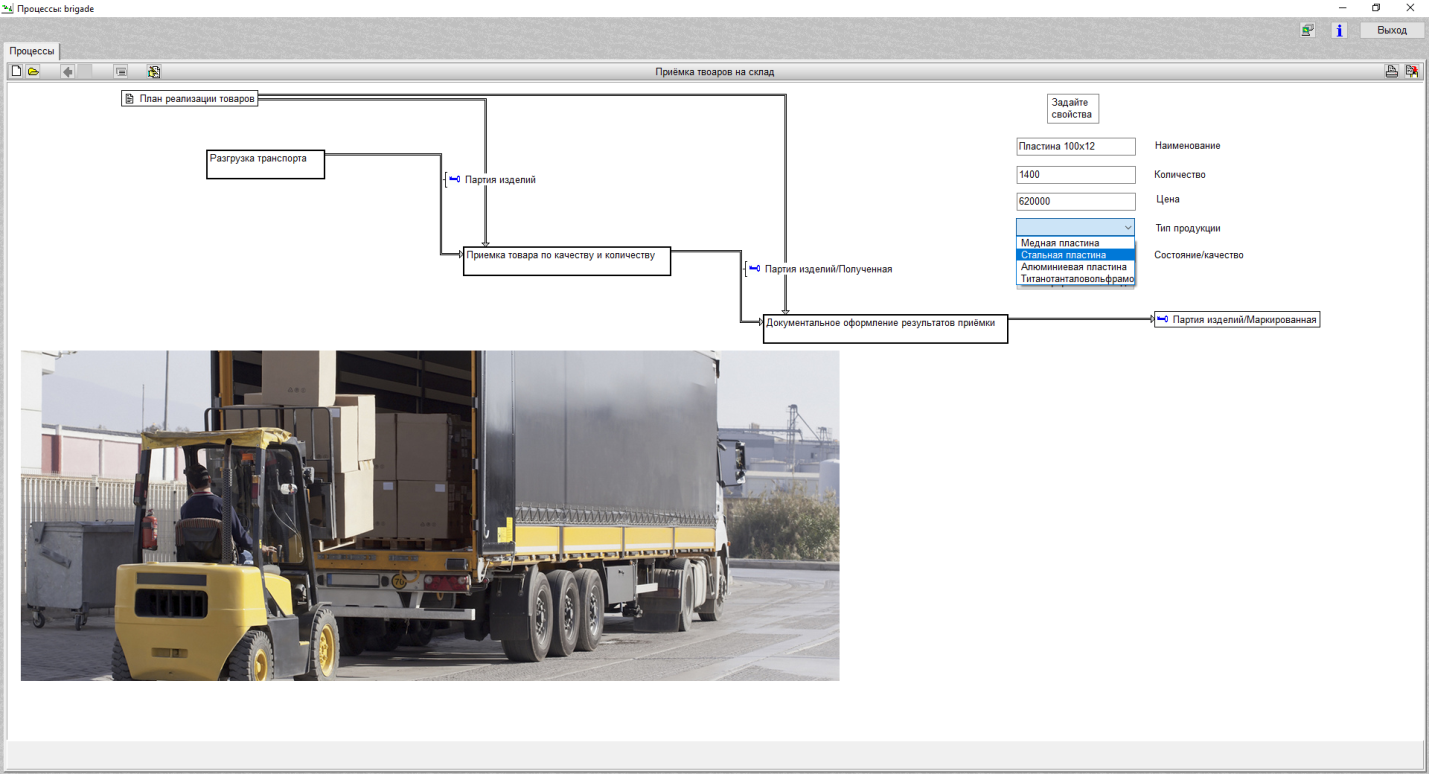


Рисунок 5 – Процесс «Приёмка товаров на склад»

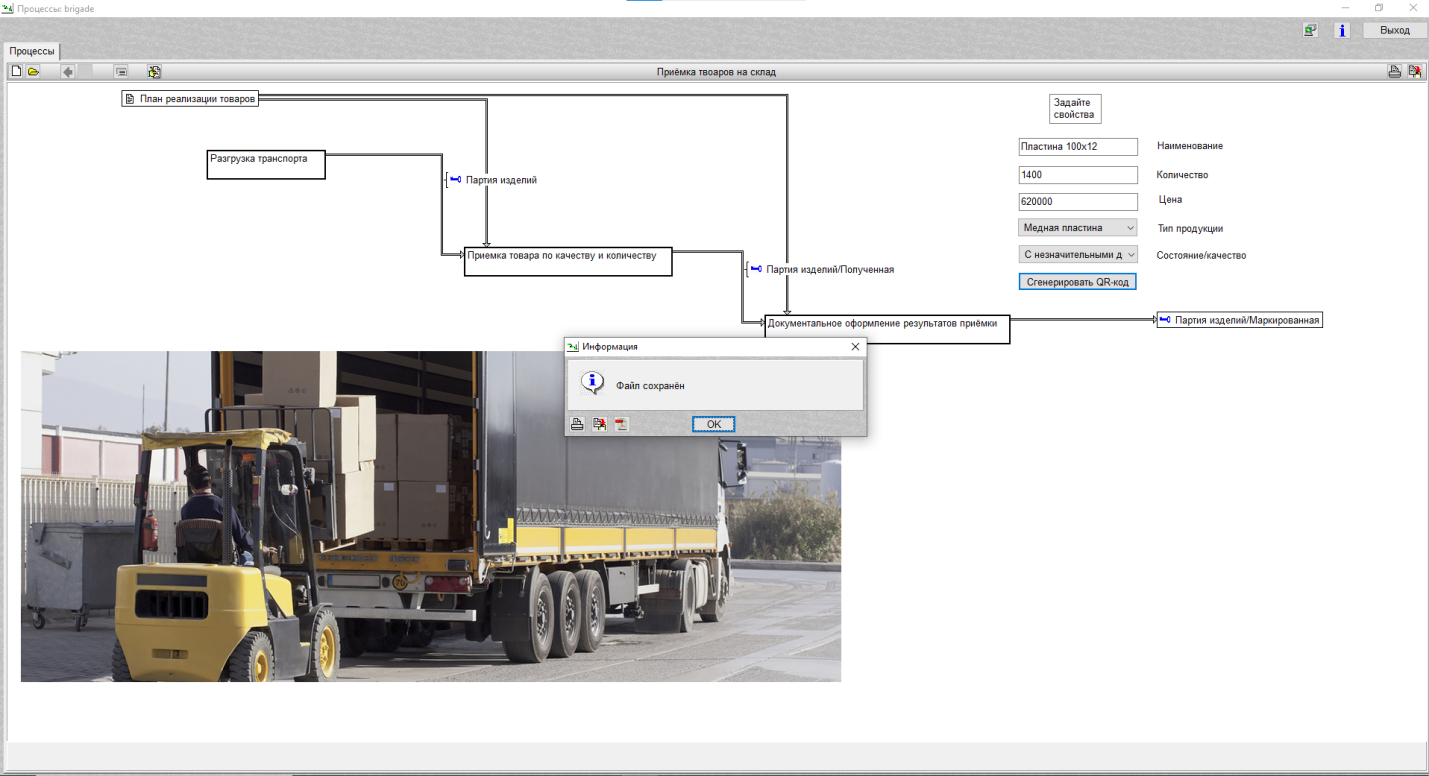


Рисунок 6 – Процесс «Приёмка товаров на склад».

3) На уровне процесса «Сортировка» подпроцесс «Формирование критериев размещения» формирует объект «Критерии». В окне описываемого процесса справа видна таблица, имеющая ячейки, помеченные разными цветами, и числа, размещенные в соответствующих ячейках. В данном случае числа – это условные позиции товаров по их оборачиваемости. Например, товары под номером 1 и 96 – самые быстро и медленно продающиеся товары на складе соответственно. Изначально товары размещены произвольным образом (рисунок 7), однако после нажатия кнопки «Сортировка», товары разместились от нижнего левого угла к верхему правому (рисунок 8). Чтобы понять почему так происходит, необходимо рассмотреть упрощенную схему склада (рисунок 9).

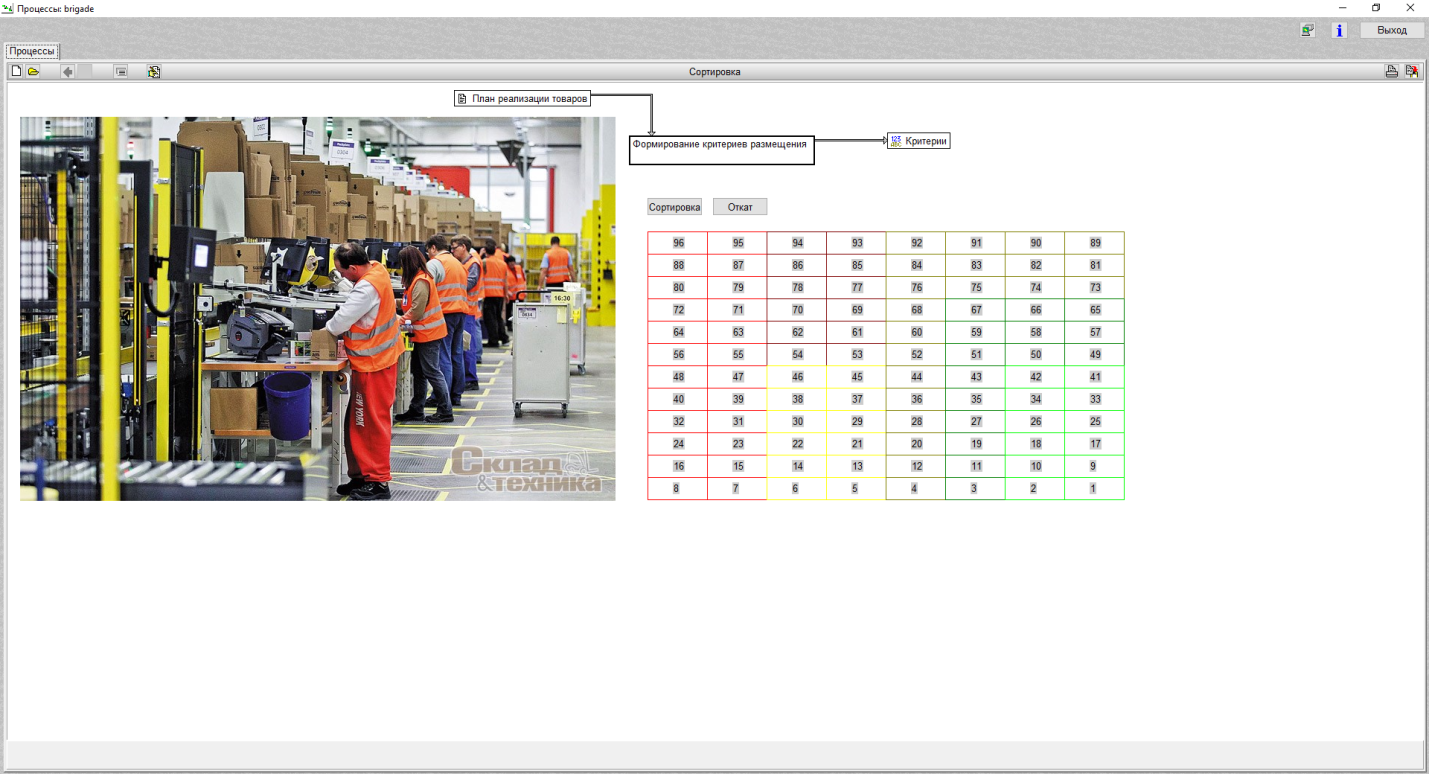


Рисунок 7 – Процесс «Сортировка»

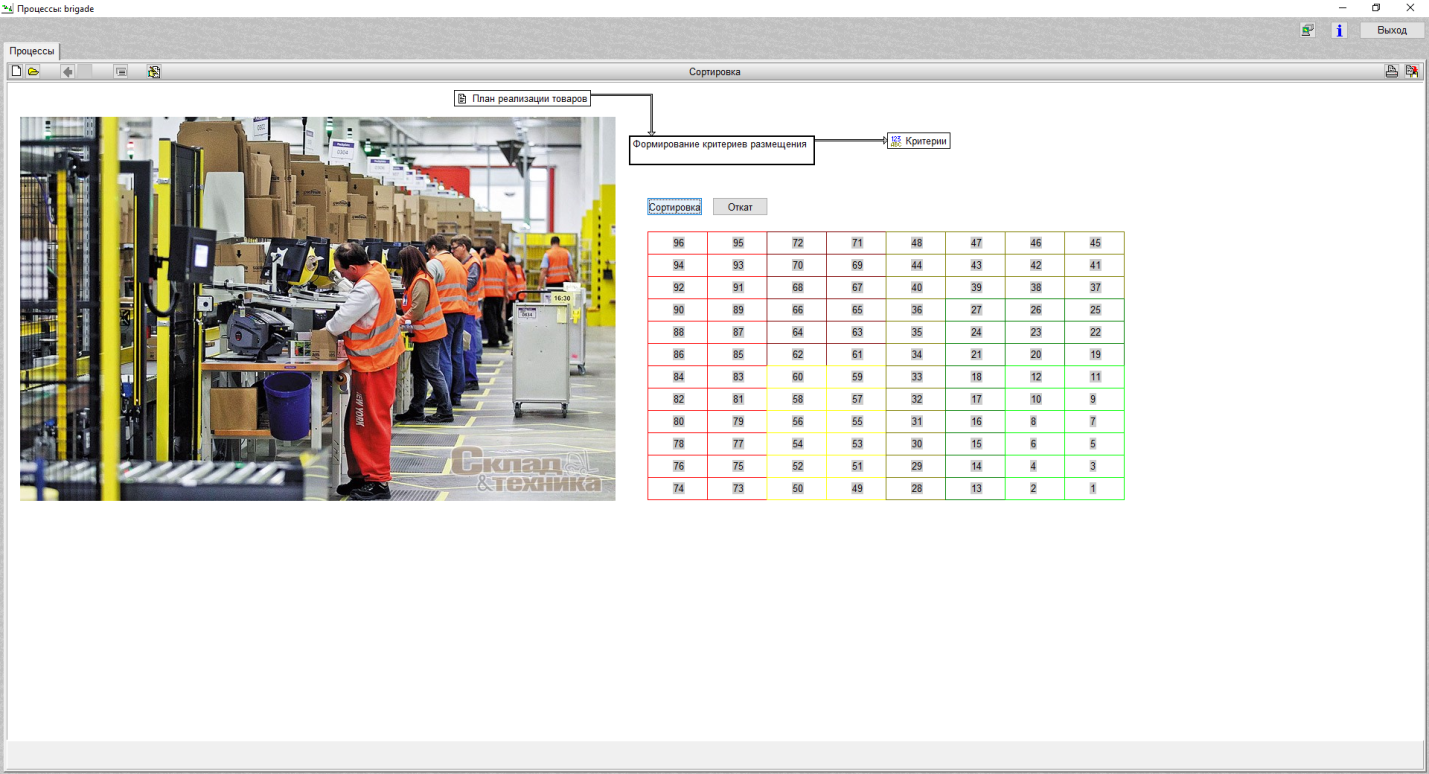


Рисунок 8 – Процесс «Сортировка»



Рисунок 9 – Простая схема склада

Необходимо заметить, что основные ячейки самых популярных товаров, независимо от их товарной категории, располагаются ближе к зоне отгрузки. Основные ячейки медленно оборачивающейся продукции отнесены на максимальное расстояние от этой зоны. Чем чаще обращаются к товарной позиции, тем ближе её необходимо замещать к зоне отгрузки и приёмки.

Функционала программы было недостаточно, чтобы написать полноценный алгоритм сортировки по различным критериям, однако описанный выше принцип был продемонстрирован на созданной таблице. Язык программы практически идентичен языку С, что позволило частично реализовать и продемонстрировать элементы интерфейса.

В процессе «Размещение» (рисунок 10) объект «Партия изделий» управляет подпроцессом «Считывание данных о товаре» и является входом подпроцесса «Размещение на соответствующем месте». Этим же подпроцессом управляет объект «Критерии», а выходом является следующая фаза «Партии изделий».

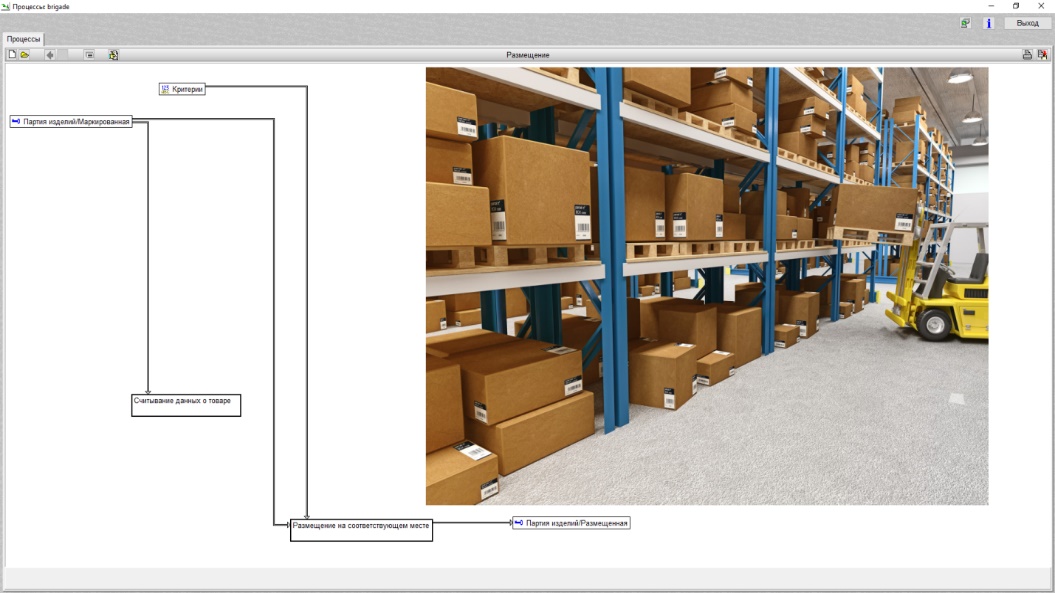


Рисунок 10 – Процесс «Размещение»

5) Последним процессом является «Отгрузка со склада», а подпроцессы, включенные в него, показаны на рисунке 11.

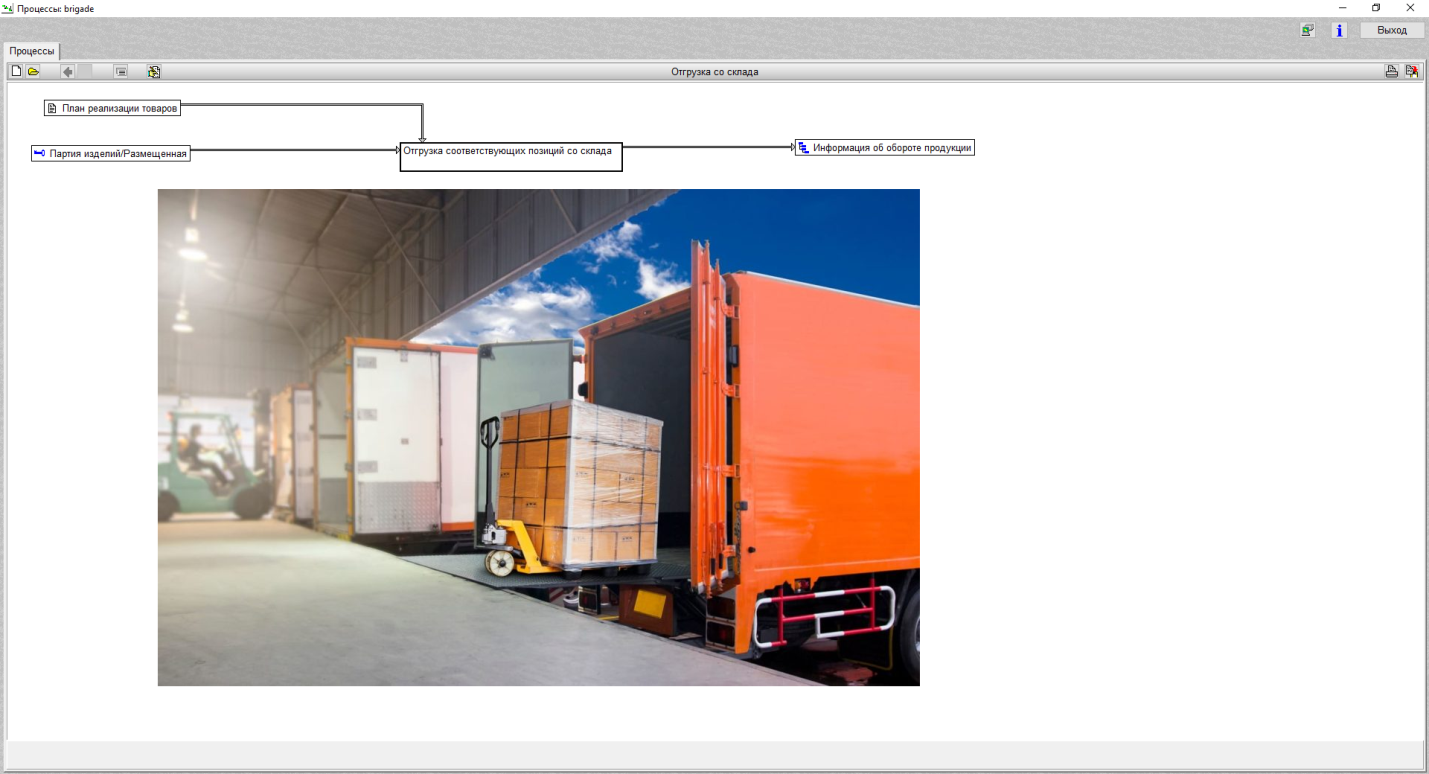


Рисунок 11 – Процесс «Размещение»

В результате создания виртуальной модели было получено более полное представление о происходящих в складских помещениях процессах. На основе информации о структуре работы складов и анализа рынка на наличие бесплатного ПО для ведения учета товаров и управления складированием продукции, было решено разработать собственный бесплатный вариант программы. В отличие от большинства найденных ПО, в системе управления складированием продукции нет бесплатного периода, доступа по подписке, ограниченного функционала и прочего, что могло бы потребовать оплату за пользование программой. Более того, большинство коммерческих ПО ориентированы на коммерческий бизнес, деловая деятельность которого сфокусирована на проведении торговых, торгово-закупочных и торгово-посреднических операций. В разработанной программе функционал максимально упрощен и не содержит лишних для материального производства элементов.

Таким образом, СУСП отлично подойдет для маленьких предприятий, не требующих обширного функционала, а также не имеющих огромной разновидности продукции и большого бюджета.

**Выводы по разделу**

В разделе были рассмотрены основные принципы складирования, описаны происходящие в складских помещениях процессы на основе созданной виртуальной модели в программе “Система планирования ресурсов и сопровождения заказов”.

# 2 ОПИСАНИЕ ВАЖНЕЙШИХ ЭТАПОВ РАЗРАБОТКИ

## 2.1 Обзор и выбор технологии доступа к данным из БД

Поскольку большую часть функционала СУСП составляет работа с базами данных, целесообразно уделить этому особое внимание.

В приложениях под ОС Windows, работающих с БД, программист может использовать различные технологии доступа к данным. Наиболее популярными из них являются: ODBC, DAO, RDO, ADO и OLE DB, BDE.

На выбор конкретной технологии наибольшее влияние оказывают следующие факторы: функциональность, легкость программирования, удобство сопровождения и производительность. Иногда необходимо получить доступ к данным "аккуратно", в другой же раз требуется скорость работы с данными. Требования удобства и простоты развертывания и сопровождения программного обеспечения также очевидны. Следует также обратить внимание и на возможность работы с одними и теми же данными нескольких пользователей – система должна поддерживать несколько подключений к БД без ощутимого уменьшения производительности.

На сегодняшний день все более широкое распространение получают OLE DB и ADO.

Эти механизмы включены в набор технологий MDAC, который поставлялся в составе ОС Microsoft Windows 2000 и Windows XP, а также в программных продуктах Microsoft Office 2000, Microsoft Office XP, Microsoft Internet Explore 5.0 и 6.0, реализующих стратегию UDA. В то же время эти механизмы доступа к базам данных соответствуют перечисленным выше требованиям [4].

UDA представляет собой «стратегию обеспечения доступа ко всем типам информации, используемой в масштабах предприятия. Она обеспечивает высокопроизводительный доступ к различным информационным источникам (как реляционные, так и нереляционные данные), в том числе к базам данных мэйнфреймов ISAM/VSAM, а также к иерархическим базам данных; файлам электронной почты и файловой системе; текстовым, графическим и географическим данным и так далее».

Помимо ADO и OLE DB в архитектуру UDA входит и ODBC, уже давно ставший стандартом Microsoft для доступа к реляционным данным и использующийся для обеспечения совместимости с более ранними разработками. Набор элементов и схема взаимодействия приложения с данными представлены на рисунке 12.

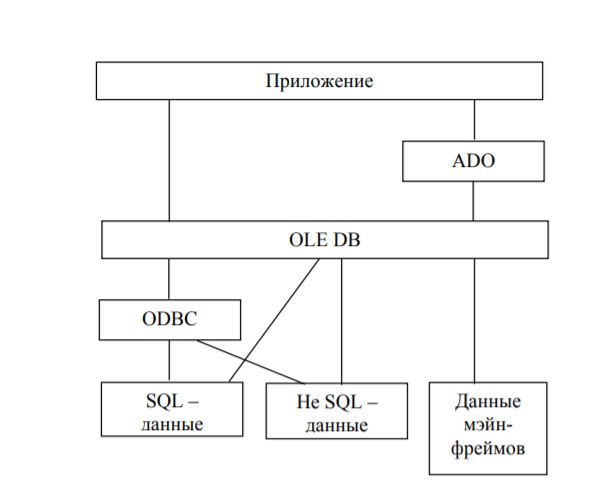


Рисунок 12 – Архитектура Microsoft Universal Data

OLE DB представляет собой интерфейс системного уровня, обеспечивающий доступ к разным источникам данных, отгорождая приложение от вида источника. OLE DB определяет набор интерфейсов компонентной объектной модели (COM), включающих в себя службы различных СУБД для обеспечения универсального доступа к данным.

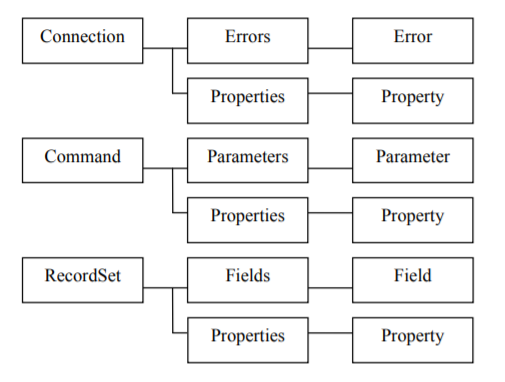
На самом высоком уровне можно выделить три главных компоненты OLE DB (рисунок 13). Это потребители (consumers), провайдеры данных (data providers) и провайдеры сервисов (service provider).



Рисунок 13 – Компоненты OLE DB

Любая компонента ПО, которая использует интерфейсы OLE DB, является потребителем OLE DB. Это может быть объектная модель ADO, инструментальное средство разработки ПО, бизнес-приложение.

Как уже было сказано, ADO представляет собой прикладной интерфейс программирования, подключенный к низкоуровневым интерфейсам OLE DB. ADO позволяет осуществлять доступ и манипулировать данными посредством любого провайдера OLE DB. ADO содержит набор объектов, используемых для соединения с источниками данных, а также для чтения, добавления, обновления и удаления данных. На рисунке 14 представлена объектная модель ADO.

  
Рисунок 14 – Объектная модель ADO

Embarcadero C++ Builder позволяет очень удобно работать с БД, и выбор был остановлен на компонентах интерфейса ADO из-за их простоты и удобства использования [5].

ADO - интерфейс программирования приложений для доступа к данным, разработанный компанией Microsoft (MS Access, MS SQL Server) и основанный на технологии компонентов ActiveX. ADO позволяет представлять данные из разнообразных источников (реляционных баз данных, текстовых файлов и т. д.) в объектно-ориентированном виде.

ADO была разработана для архитектуры клиент-сервер. Эта архитектура в течение ряда лет была достаточно прогрессивной, если учесть, что пришла она на смену архитектуре файл-сервер.

Объекты в технологии ADO являются базой для построения «приближенных к программисту» компонент C++ Builder. Механизм взаимодействия клиентского приложения, создаваемого в среде C++ Builder, с базой данных через ADO осуществляется с помощью компонент, помещенных на вкладке ADO панели инструментов.

Объекты БД и компоненты, поддерживающие ADO и работающие с БД, связаны с визуальными компонентами для работы с данными программной цепочкой, приведенной на рисунке 15. Из этой цепочки видно, что механизм ADO в C++ Builder многое позаимствовал из механизма доступа BDE.

Иерархия VCL-компонент ADO в C++ Builder соответствует иерархии компонент ADO от Microsoft, также эти VCL-компоненты, помимо свойств компонент ADO, инкапсулируют механизм доступа к данным, являясь наследниками более общих классов C++ Builder для работы с БД (рисунок 16).

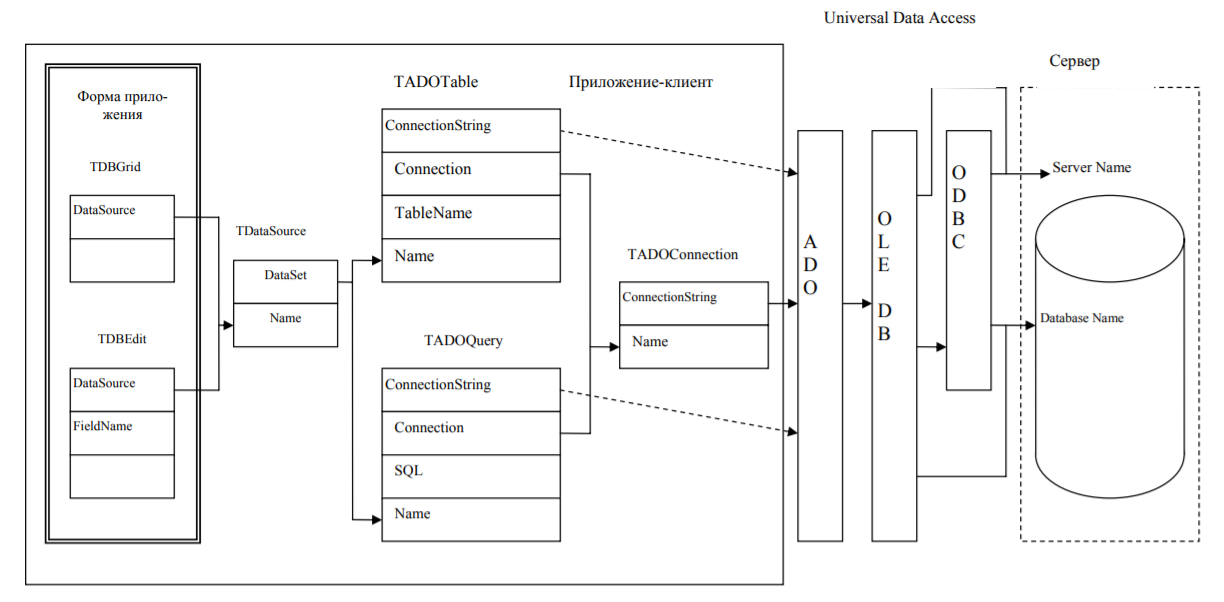


Рисунок 15 - Связь визуальных компонент экранной формы с объектами БД

****

Рисунок 16 - Иерархия компонент ADO в C++ Builder

Пользуясь компонентами TADOTable, TADOConnection и компонентом TDataSource, который является посредником между компонентами TDataSet и компонентами Data Controls, в СУСП была создана БД сотрудников предприятия, имеющих доступ ко второй БД с набором характеристик продукции, хранящейся на складе [6].

## 2.2 Описание работы алгоритма генерации QR-кодов

Еще одной важной составляющей СУСП является генерация QR-кодов для маркировки изделий. Для реализации такой возможности необходимо вникнуть в саму структуру и алгоритм генерации QR-кодов.

Задумка шифровать информацию с помощью графических элементов, которые будут распознаваться автоматически, появилась еще в середине двадцатого века. Создателя первого штрихового кода Нормана Вудланда вдохновила азбука Морзе. Из-за сложности реализации замысла на тот момент прошло двадцать лет до появления первого товара со штрихкодом.

Штрихкоды оказались очень удобными и быстро появились в промышленности. Однако для некоторых компаний этого было недостаточно. В японской Denso Wave, работники просили руководство разработать более эффективную систему, и в 1994 она появилась — в виде матричного (двумерного) кода. Его назвали QR Code — код быстрого реагирования. QR-код позволяет кодировать во много раз больше информации, чем штрихкоды, легко распознается сканирующим оборудованием и может быть прочитан даже при повреждении [7].

Процесс генерации QR кода делится на несколько чётких шагов:

1. Кодирование данных.
2. Добавление служебной информации и заполнения.
3. Разделение информации на блоки.
4. Создание байтов коррекции.
5. Объединение блоков.
6. Размещение информации на QR коде.

**Кодирование данных.** QR код позволяет кодировать данные несколькими способами, в зависимости от того, какие символы используются: цифровое, буквенно-цифровое, кандзи и побайтовое кодирование. Цифровое кодирование подразумевает использования только цифр от 0 до 9, буквенно-цифровое - прописные буквы латинского алфавита, цифры и символы $%\*+-./: и пробел. Сначала создается пустая последовательность бит, которая будет заполняться в дальнейшем в зависимости от типа кодирования.

В случае цифрового кодирования, требуется 10 бит на 3 символа. Вся последовательность символов разбивается в группы по 3 цифры, где каждая группа переводится в 10-битное двоичное число с добавлением бита к последовательности. Если общее количество символов не кратно 3, то в случае, когда в конце остаётся 2 символа, полученное двузначное число кодируется 7 битами, в ином же случае - 4 битами.

В случае буквенно-цифрового кодирования на каждые 2 символа требуется 11 бит информации. Входной поток символов разбивается в группы по 2, где в группе каждый символ кодируется согласно таблице, показанной на таблице 1. Значение первого символа в группе умножается на 45 и суммируется со значением второго символа. Полученное число переводится в 11-битное двоичное число и добавляется с добавлением бита к последовательности. Если в последней группе 1 символ, то его значение кодируется 6-битным числом с добавлением бита к последовательности.

Таблица 1 – Значения символов в буквенно-цифровом кодировании

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| U | V | W | X | Y | Z | Пробел | $ | % | \* | + | - | . | / | : |
| 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 |

Побайтовый способ кодирования является универсальным и позволяет закодировать любые символы. Единственным его недостатком является относительно низкое количество передаваемой информации. В этом случае текст кодируется в любой кодировке, и полученная последовательность байт берётся в неизменном виде.

**Добавление служебной информации.** На этом этапе необходимо задать уровень коррекции: чем он выше, тем выше допустимый уровень повреждения изображения и тем меньше информации при равном размере. Всего есть 4 уровня коррекции: L (допустимо максимум 7% повреждений), M (15%), Q (25%) и H (30%). Чаще всего используется уровень M.

Ещё одно свойство QR кода — его версия. Номера версий существуют от 1 до 40. Каждая версия имеет особенности в конфигурации и количестве точек(модулей) составляющих QR-код. Версия 1 содержит 21×21 модулей, версия 40 — 177×177. От версии к версии размер кода увеличивается на 4 модуля на сторону. От уровня версии и уровня коррекции зависит количество кодируемой информации. В таблице 2 указано максимальное общее количество полезной информации в битах, которое можно закодировать в QR коде этой версии.

Таблица 2 – Характеристики различных версий QR-кодов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Версия | Количество модулей | Уровень коррекции ошибок | Максимальное количество символов с учетом уровня коррекции ошибок и типа символов | | | |
|  |
| Числа: 0 — 9 | Числа и символы латинского алфавита \*, пробел, $ % \* + — . / : | Двоичные данные | Символы японского алфавита Kanji |  |
|  |
|  |
| 1 | 21×21 | L | 41 | 25 | 17 | 10 |  |
| M | 34 | 20 | 14 | 8 |  |
| Q | 27 | 16 | 11 | 7 |  |
| H | 17 | 10 | 7 | 4 |  |
| 2 | 25×25 | L | 77 | 47 | 32 | 20 |  |
| M | 63 | 38 | 26 | 16 |  |
| Q | 48 | 29 | 20 | 12 |  |
| H | 34 | 20 | 14 | 8 |  |
| 4 | 33×33 | L | 187 | 114 | 78 | 48 |  |
| M | 149 | 90 | 62 | 38 |  |
| Q | 111 | 67 | 46 | 28 |  |
| H | 82 | 50 | 34 | 21 |  |
| 10 | 57×57 | L | 652 | 395 | 271 | 167 |  |
| M | 513 | 311 | 213 | 131 |  |
| Q | 364 | 221 | 151 | 93 |  |
| H | 288 | 174 | 119 | 74 |  |
| 40 | 177×177 | L | 7,089 | 4,296 | 2,953 | 1,817 |  |
| M | 5,596 | 3,391 | 2,331 | 1,435 |  |
| Q | 3,993 | 2,42 | 1,663 | 1,024 |  |
| H | 3,057 | 1,852 | 1,273 | 784 |  |

После определения уровня коррекции и версии QR кода, необходимо добавить 2 поля перед последовательностью бит, полученной ранее. Первое поле - способ кодирования, а второе - количество данных. Способ кодирования – это поле длиной 4 бита, которое имеет следующие значения: 0001 для цифрового кодирования, 0010 для буквенно-цифрового и 0100 для побайтового. Количество данных - количество кодируемых символов или байт для побайтового кодирования, представленное двоичным числом определённой длины. Длина этого числа определяется по таблице 3.

Таблица 3 – Длина поля количества данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | Версия 1–9 | Версия 10–26 | Версия 27–40 |
| Цифровое | 10 бит | 12 бит | 14 бит |
| Буквенно-цифровое | 9 бит | 11 бит | 13 бит |
| Побайтовое | 8 бит | 16 бит | 16 бит |

К этапу заполнения уже имеется последовательность бит данных, количество бит в которой может не равняться 8. В этом случае необходимо дополнить нулями пустые биты до 8. Далее последовательность разбивается на группы по 8 бит, чтобы представить ее в виде последовательности байт. Если количество бит в полученной последовательности байт меньше необходимого для выбранной версии, то её следует дополнить чередующимися байтами 11101100 и 00010001 [8].

**Разделение информации на блоки.** Последовательность байт, полученная ранее, (далее данные) разделяется на определённое для версии и уровня коррекции количество блоков, которое приведено в таблице 4. Если количество блоков равно одному, то этот этап пропускается.

Таблица 4 – Количество блоков

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Версия | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| L | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 4 |
| M | 1 | 1 | 1 | 2 | 2 | 4 | 4 | 4 | 5 | 5 |
| Q | 1 | 1 | 2 | 2 | 4 | 4 | 6 | 6 | 8 | 8 |
| H | 1 | 1 | 2 | 4 | 4 | 4 | 5 | 6 | 8 | 8 |
| Версия | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| L | 4 | 4 | 4 | 4 | 6 | 6 | 6 | 6 | 7 | 8 |
| M | 5 | 8 | 9 | 9 | 10 | 10 | 11 | 13 | 14 | 16 |
| Q | 8 | 10 | 12 | 16 | 12 | 17 | 16 | 18 | 21 | 20 |
| H | 11 | 11 | 16 | 16 | 18 | 16 | 19 | 21 | 25 | 25 |
| Версия | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| L | 8 | 9 | 9 | 10 | 12 | 12 | 12 | 13 | 14 | 15 |
| M | 17 | 17 | 18 | 20 | 21 | 23 | 25 | 26 | 28 | 29 |
| Q | 23 | 23 | 25 | 27 | 29 | 34 | 34 | 35 | 38 | 40 |
| H | 25 | 34 | 30 | 32 | 35 | 37 | 40 | 42 | 45 | 48 |
| Версия | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| L | 16 | 17 | 18 | 19 | 19 | 20 | 21 | 22 | 24 | 25 |
| M | 31 | 33 | 35 | 37 | 38 | 40 | 43 | 45 | 47 | 49 |
| Q | 43 | 45 | 48 | 51 | 53 | 56 | 59 | 62 | 65 | 68 |
| H | 51 | 54 | 57 | 60 | 63 | 66 | 70 | 74 | 77 | 81 |

Для определения количества байт в каждом блоке, необходимо разделить общее число байт на количество блоков данных. При получении нецелого числа, остаток от деления покажет количество блоков, в которых будет содержаться на 1 байт больше, чем в остальных. Такими дополненными блоками будут являться последние блоки. Блоки заполняются байтами данных полностью, и при заполнении первого блока, очередь переходит к следующему.

**Создание байтов коррекции.** К каждому блоку применяется алгоритм, основанный на алгоритме Рида–Соломона [9]. Сначала определяется количество создаваемых байтов коррекции по таблице 5.

Таблица 5 – Количество байтов коррекции на один блок

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Версия | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| L | 7 | 10 | 15 | 20 | 26 | 18 | 20 | 24 | 30 | 18 |
| M | 10 | 16 | 26 | 18 | 24 | 16 | 18 | 22 | 22 | 26 |
| Q | 13 | 22 | 18 | 26 | 18 | 24 | 18 | 22 | 20 | 24 |
| H | 17 | 28 | 22 | 16 | 22 | 28 | 26 | 26 | 24 | 28 |
| Версия | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| L | 20 | 24 | 26 | 30 | 22 | 24 | 28 | 30 | 28 | 28 |
| M | 30 | 22 | 22 | 24 | 24 | 28 | 28 | 26 | 26 | 26 |
| Q | 28 | 26 | 24 | 20 | 30 | 24 | 28 | 28 | 26 | 30 |
| H | 24 | 28 | 22 | 24 | 24 | 30 | 28 | 28 | 26 | 28 |
| Версия | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| L | 28 | 28 | 30 | 30 | 26 | 28 | 30 | 30 | 30 | 30 |
| M | 26 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 |
| Q | 28 | 30 | 30 | 30 | 30 | 28 | 30 | 30 | 30 | 30 |
| H | 30 | 24 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| Версия | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| L | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| M | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 |
| Q | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| H | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |

По количеству байт коррекции определяется генерирующий многочлен (таблица 6).

Таблица 6 – Генерирующие многочлены

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Количество байтов коррекции | 7 | 10 | 13 | 15 | 16 | 17 | 18 | 20 | 22 | 24 | 26 | 28 | 30 |
| Генерирующий многочлен | 87, 229, 146, 149, 238, 102, 21 | 251, 67, 46, 61, 118, 70, 64, 94, 32, 45 | 74, 152, 176, 100, 86, 100, 106, 104, 130, 218, 206, 140, 78 | 8, 183, 61, 91, 202, 37, 51, 58, 58, 237, 140, 124, 5, 99, 105 | 120, 104, 107, 109, 102, 161, 76, 3, 91, 191, 147, 169, 182, 194, 225, 120 | 43, 139, 206, 78, 43, 239, 123, 206, 214, 147, 24, 99, 150, 39, 243, 163, 136 | 215, 234, 158, 94, 184, 97, 118, 170, 79, 187, 152, 148, 252, 179, 5, 98, 96, 153 | 17, 60, 79, 50, 61, 163, 26, 187, 202, 180, 221, 225, 83, 239, 156, 164, 212, 212, 188, 190 | 210, 171, 247, 242, 93, 230, 14, 109, 221, 53, 200, 74, 8, 172, 98, 80, 219, 134, 160, 105, 165, 231 | 229, 121, 135, 48, 211, 117, 251, 126, 159, 180, 169, 152, 192, 226, 228, 218, 111, 0, 117, 232, 87, 96, 227, 21 | 173, 125, 158, 2, 103, 182, 118, 17, 145, 201, 111, 28, 165, 53, 161, 21, 245, 142, 13, 102, 48, 227, 153, 145, 218, 70 | 168, 223, 200, 104, 224, 234, 108, 180, 110, 190, 195, 147, 205, 27, 232, 201, 21, 43, 245, 87, 42, 195, 212, 119, 242, 37, 9, 123 | 41, 173, 145, 152, 216, 31, 179, 182, 50, 48, 110, 86, 239, 96, 222, 125, 42, 173, 226, 193, 224, 130, 156, 37, 251, 216, 238, 40, 192, 180 |

Перед выполнением цикла необходимо подготовить массив, длина которого равна максимуму из количества байтов в текущем блоке и количества байтов коррекции, и заполнить его начало байтами из текущего блока, а конец нулями. Количество итераций цикла, описанного далее, равно количеству байт данных в текущем блоке.

1. Значение первого элемента массива сохраняется в некоторой переменной и удаляется из массива со сдвигом влево, при этом последний элемент заполняется нулём.
2. Если значение переменной равно нулю, то происходит переход к следующей итерации цикла.
3. Ищется соответствующее значению переменной число из значений для обратного поля Галуа длиной 256 и заносится во вторую переменную
4. Далее для N первых элементов, где N - количество байтов коррекции, i - счётчик цикла:
   1. К i-му значению генерирующего многочлена необходимо прибавить значение второй переменной и записать эту сумму в неё же.
   2. Если полученное значение больше 254, используется его остаток при делении на 255.
   3. Ищется значение для поля Галуа, соответствующее полученному ранее значению, и проводится операция побитового сложения по модулю 2 с i-м значением подготовленного массива. Записывается полученное значение в i-ю ячейку подготовленного массива.

Первые N байтов подготовленного массива после цикла - байты коррекции. Для каждого блока данных получается соответствующий блок байтов коррекции.

**Объединение блоков**. На данном этапе имеется несколько блоков данных и столько же блоков байтов коррекции, которые необходимо объединить в один поток байт. Для этого из каждого блока данных по очереди берётся один байт информации, когда очередь доходит до последнего блока, из него берётся байт и очередь переходит к первому блоку. Так продолжается до тех пор, пока в каждом блоке не кончатся байты. Если в текущем блоке уже нет байт, то он пропускается. Аналогичным образом надо сделать с блоками байтов коррекции. Они берутся в том же порядке, что и соответствующие блоки данных.

**Размещение информации на QR коде.** Последняя последовательность байт является готовой для помещения на холст, который состоит из модулей – элементарных квадратов. На рисунке 17 показан пример холста со всеми базовыми элементами.

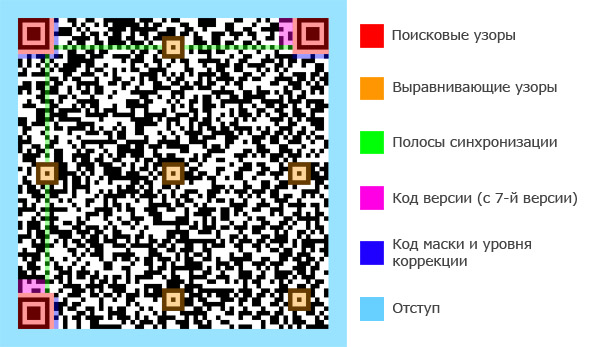


Рисунок 17 – Структура QR-кода

Размер QR кода зависит только от версии. 3 чёрных квадрата по углам размером 3 на 3 модуля, которые окружены рамкой из белых модулей, которая окружена рамкой из чёрных модулей, которая окружена рамкой из белых модулей только с тех сторон, где нет отступа, называются поисковыми узорами. Начиная со 2-й версии, на холсте появляются чёрные квадраты размером 1 на 1 модуль, которые окружены рамкой из белых модулей, которая окружена рамкой из чёрных модулей. Они называются выравнивающими узорами. Полосы синхронизации начинаются от самого нижнего правого чёрного модуля верхнего левого поискового узора и идут, чередуя чёрные и белые модули, вниз и вправо до противоположных поисковых узоров. Все эти элементы нужны для корректного восприятия QR-кода в пространстве устройством.

Начиная с седьмой версии используется код версии. Он зеркально дублируется в двух местах рядом с поисковыми узорами. Модули строятся согласно таблице 7.

Таблица 7 – Коды версий

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7 | 8 | 9 | 10 | 11 | 12 |
| 10011110100110 | 10 001 011 100 111 000 | 110 111 011 000 000 000 | 101 001 111 110 000 000 | 1 111 111 010 111 100 | 1 101 100 100 011 010 |
| 13 | 14 | 15 | 16 | 17 | 18 |
| 101011100000100000 | 110 101 000 110 100 000 | 10 011 000 010 011 100 | 11 100 010 001 011 100 | 111 010 010 101 100 000 | 100 100 110 011 100 000 |
| 19 | 20 | 21 | 22 | 23 | 24 |
| 10110111011000 | 000000 101001 111110 | 100 110 101 101 000 000 | 111 000 001 011 000 000 | 11 110 001 111 111 000 | 1 101 001 101 100 100 |
| 25 | 26 | 27 | 28 | 29 | 30 |
| 101011001001011000 | 110 101 101 111 011 000 | 10 011 101 011 100 000 | 10 001 110 101 000 100 | 110 111 110 001 111 000 | 101 001 010 111 111 000 |
| 31 | 32 | 33 | 34 | 35 | 36 |
| 1111010011000010 | 101 000 011 000 101 000 | 1 110 011 100 010 000 | 10 000 111 010 010 100 | 110 110 111 110 101 000 | 110 100 100 000 001 000 |
| 37 | 38 | 39 | 40 |
| 10010100100110000 | 1 100 000 010 110 110 | 101 010 000 110 001 000 | 111 001 000 100 010 000 |

Код маски и уровня коррекции также дублируется в 2-х местах: рядом с верхним левым поисковым узором и рядом с нижним и правым поисковыми узорами. В нём особым образом зашифрованы код маски и код уровня коррекции.

Всё оставшееся свободное пространство на холсте разбивается на столбики: каждые 2 модуля, не важно, что находится в этих модулях, кроме вертикальной полосы синхронизации, которая просто пропускается. Заполнение начинается с правого нижнего угла, идёт в пределах столбика справа налево, снизу вверх с наложением одной из восьми масок на каждый модуль. Если текущий модуль занят (например, полосой синхронизации или выравнивающим узором), то он просто пропускается. Если достигнут верх столбика, то движение продолжается с верхнего правого угла столбика, который расположен левее, и идёт сверху вниз. Достигнув низа, движение продолжается от нижнего правого угла столбика, который расположен левее, и идёт снизу вверх. И так далее, пока всё свободное пространство не будет заполнено. Визуально этот процесс представлен на рисунке 18.

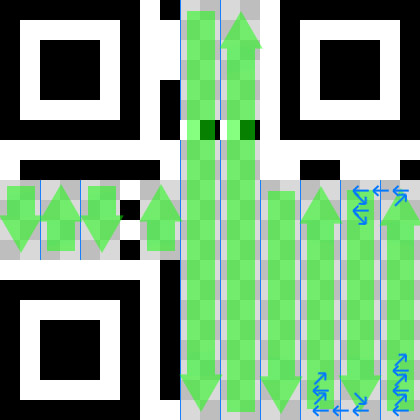


Рисунок 18 – Заполнение QR-кода

Описанный выше алгоритм генерации QR-кодов практически универсальный и уже используется во многих библиотеках, отсекающих надобность писать низкоуровневый код, тем самым упрощающих создание QR-кода. В СУСП была использована библиотека libqrencode, разработанная Фукучи Кентаро в 2006 году. В настоящее время эта библиотека портирована на языки Python, Python ctypes, Haskell, Ruby, PHP, OCaml, Crystal, Eiffel, Lua. Эта библиотека является свободным программным обеспечением, которую можно распространять и/или изменять в соответствии с условиями GNU Lesser General Public License, опубликованными Фондом свободного программного обеспечения.

**Выводы по разделу**

В разделе были представлены возможные технологии доступа к данным, был обоснован выбор компонентов ADO. Был описан алгоритм генерации QR-кодов, заложенный в используемую в разработке библиотеку libqrencode.

# 3 **РЕЗУЛЬТАТЫ РАЗРАБОТКИ**

## 3.1 Описание работы программы

СУСП представляет собой программу, в которой содержится 7 окон, 3 из которых являются основными, а остальные 4 функциональными. Общая структура программы представлена на рисунке 19, где прямоугольниками обозначены модули, которые для удобства были названы идентично содержащимся в них формам. Шестиугольниками обозначены файлы баз данных Microsoft Access.

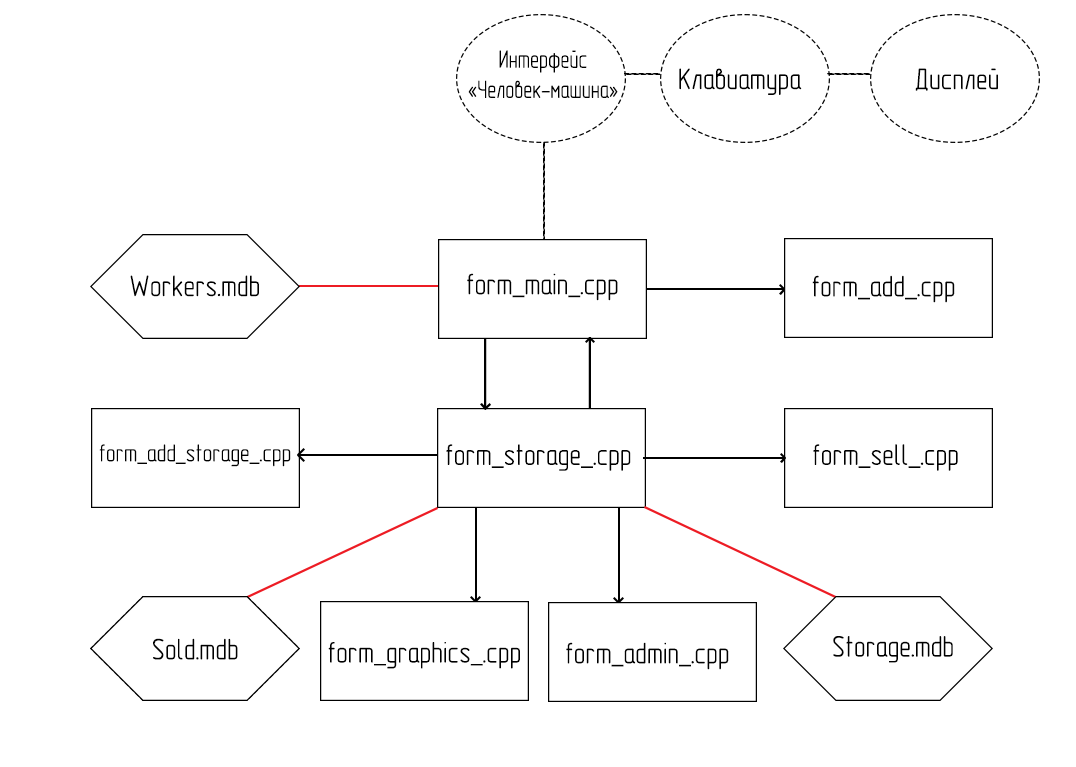


Рисунок 19 – Схема структуры программы

База данных в СУСП представляет собой 3 таблицы:

1. Workers – таблица базы данных сотрудников предприятия, которая имеет 3 поля: № (id), ФИО и КОД. Первое поле представляет собой уникальный номер для каждого сотрудника, второе хранит в себе его ФИО, а третье хранит пароль для доступа в программу.
2. Storage – таблица базы данных продукции на складе, которая имеет 7 полей:
   1. № (id) – уникальный номер для каждого вида товара с определенным набором характеристик.
   2. Наименование – название конкретного вида товара на складе.
   3. Тип – к какой категории принадлежит определенный вид товара.
   4. Состояние – текущее состояние партии товаров в самом широком смысле. В примере это поле подразумевает качество партии, т.е. наличие дефектов. В зависимости от производства это поле может быть использовано по-разному.
   5. Дата поступления – дата приемки товара или партии на склад.
   6. Себестоимость – оценочная стоимость единицы товара в партии.
   7. Количество – количество товаров в партии.
3. Sold – таблица базы данных реализованной продукции, которая имеет 9 полей:
   1. Id - уникальный номер для каждого вида товара с определенным набором характеристик, отправленного на отгрузку. Это поле скрыто от пользователя.
   2. № - номер для каждого вида товара с определенным набором характеристик из таблицы Storage.
   3. Наименование – название конкретного вида товара, отгружаемого со склада.
   4. Тип – к какой категории принадлежит определенный вид товара.
   5. Состояние – текущее состояние партии товаров в самом широком смысле.
   6. Дата поступления – дата приемки товара или партии на склад.
   7. Себестоимость – оценочная стоимость единицы товара в партии.
   8. Количество – количество товаров в партии на отгрузку.
   9. Дата отгрузки – дата отгрузки товара или партии со склада.

Предпоследняя таблица связана с последней по типу «один ко многим», а общая структура базы данных СУСП представлена на рисунке 20.

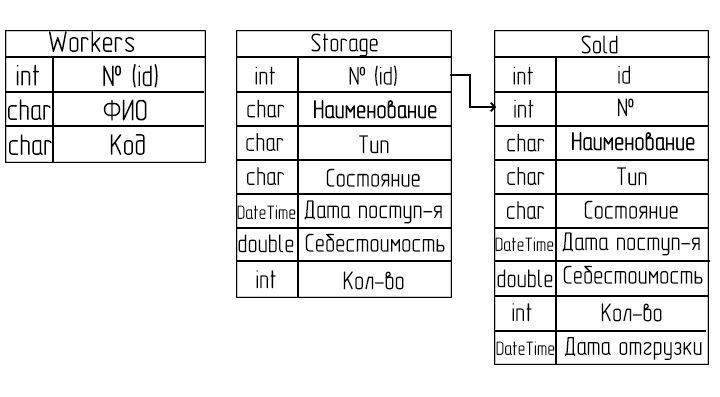


Рисунок 20 – Структура базы данных СУСП

При запуске программы, пользователя встречает окно «Вход в учетную запись сотрудника», в котором содержится список верифицированных сотрудников, имеющих доступ к работе с данными склада, уникальный номер (код) каждого сотрудника и поле для входа в учетную запись. Рядом со списком отображается общее число сотрудников. Чтобы войти в свою учетную запись СУСП, необходимо выделить из списка свою фамилию и ввести в поле для входа свой пароль. Окно и процесс входа наглядно показаны на рисунке 21.

Для появления возможности удалять, редактировать или создавать новые учетные записи сотрудников предусмотрена возможность войти с правами доступа администратора. Пароль администратора устанавливается один раз при компиляции программы и должен быть известен только управляющему склада. При необходимости изменить пароль, следует обратиться к штатному программисту, либо человеку, отвечающему за работу программы на предприятии. Также, в случае если сотрудник забыл или желает изменить свой пароль, в режиме администратора открывается доступ к списку паролей учетных записей. На рисунке 22 показано окно в режиме администратора, а на рисунке 23 форма, появляющаяся при нажатии на кнопку создания новой учетной записи сотрудника. При создании таковой, программа попросит ввести и подтвердить пароль, а также не даст оставить поля пустыми.

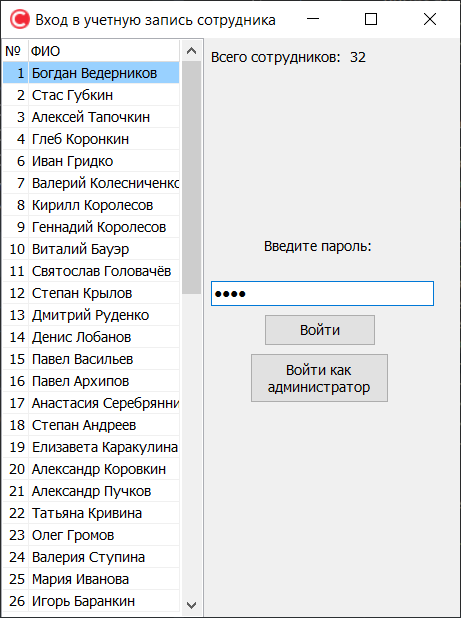


Рисунок 21 – Окно входа в учетную запись СУСП



Рисунок 22 - Окно входа в учетную запись СУСП в режиме администратора

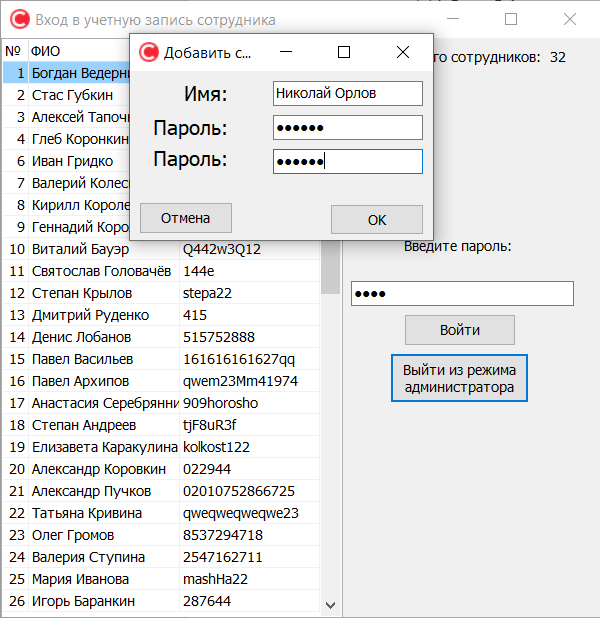


Рисунок 23 – Создание новой учетной записи сотрудника

После входа в свою учетную запись, пользователя встречает окно с таблицей учета продукции на складе и 8 кнопками на боковой панели. На рисунке 24 показано это окно, а также пронумерованы все кнопки.

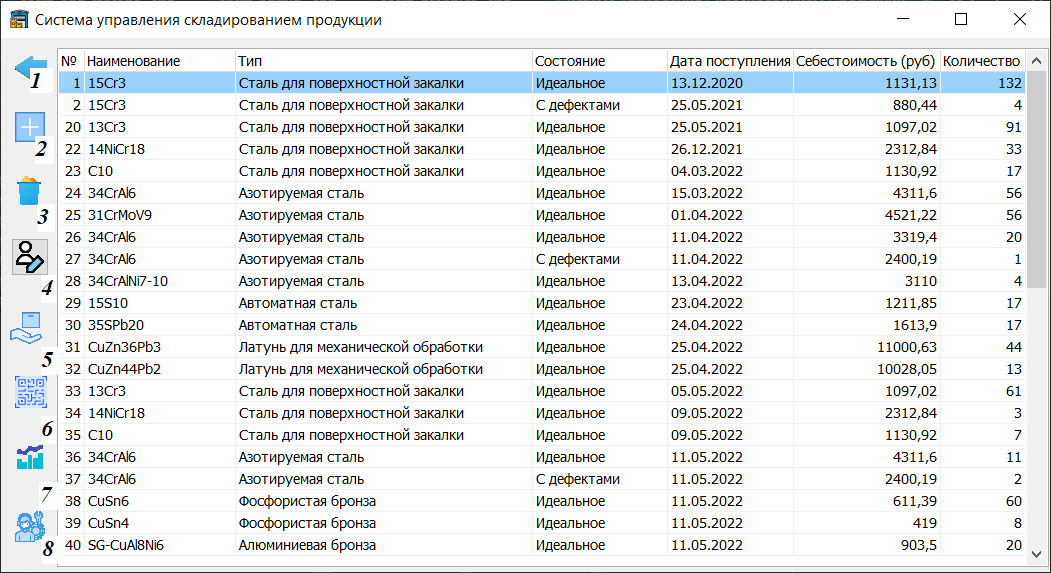


Рисунок 24 – Окно учета продукции СУСП

1. Синей стрелочкой обозначена кнопка «Назад», которая возвращает пользователя на стартовую форму.
2. Кнопка с белым плюсом позволяет добавить в БД продукцию, поступившую на склад. Окно, вызываемое при нажатии этой кнопки, показано на рисунке 25. Программа не дает оставить поля пустыми, ввести некорректные для выбранного поля символы, а также распознает, если пользователь вводит характеристики, которые уже есть в БД склада и добавляет нужное число в поле «Количество» существующей единицы, вместо добавления идентичной новой.

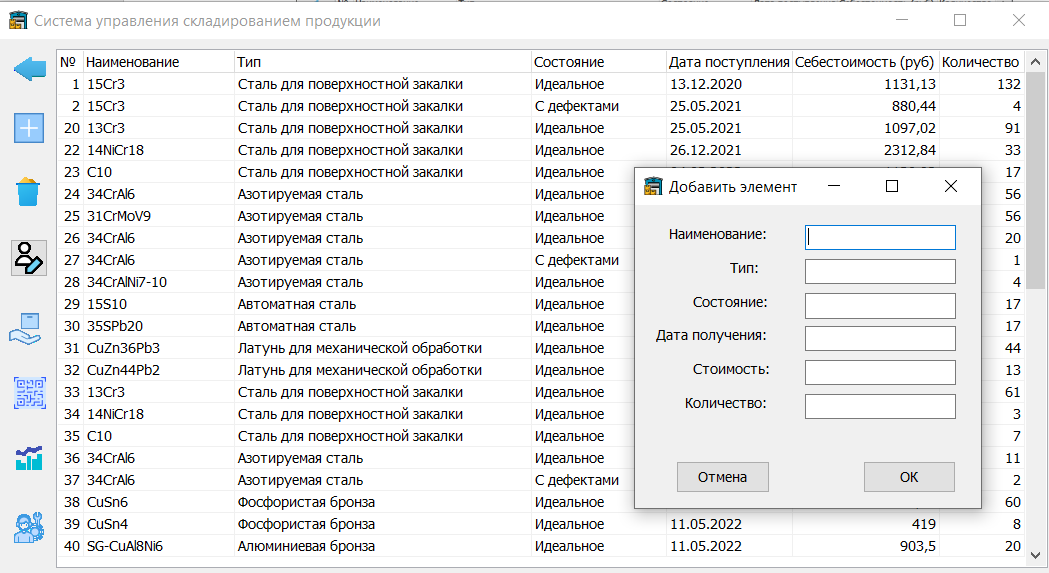


Рисунок 25 – Окно добавления продукции на склад СУСП

1. Кнопка с иконкой корзины позволяет удалить выделенную запись из БД склада.
2. Следующая кнопка позволяет перейти в режим редактирования для изменения каких-либо ячеек при необходимости.
3. Иконка с рукой предполагает отгрузку выделенной продукции. При нажатии появляется небольшое поле, куда требуется ввести количество единиц товара на отгрузку. Выбранное количество единиц товара отправляется в другую базу данных отгрузки, скрытую от обычного пользователя.
4. Кнопка со значком QR-кода генерирует набор QR-кодов в формате .bmp для каждой единицы продукции выделенного поля. Формат .bmp был выбран, потому что с ним удобно работать, и при необходимости несложно конвертировать в любой другой. В подпапке QR папки с программой создается папка с наименованием, в которой содержатся уникальные QR-кода для каждой единицы товара (рисунок 26).

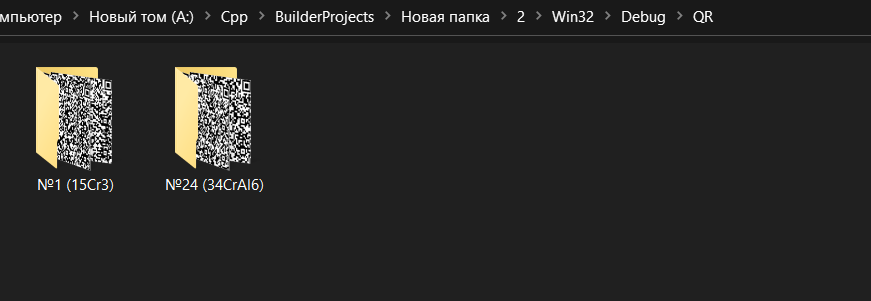


Рисунок 26 – Папка с QR-кодами

В каждый код зашифрованы все характеристики конкретной единицы продукции, разделенные пробелом: уникальный номер (код) в базе данных, наименование, тип, состояние, дата поступления на склад, себестоимость, количество единиц в партии и номер единицы внутри партии. Этой информации достаточно для полного представления о конкретном товаре, попавшем в руки сотруднику, либо для дальнейшего использования в автоматизированной системе. Например, при сканировании кода роботом для дальнейшего расположения в помещении склада. Полученный код также без проблем считывается и корректно отображается с помощью обычного смартфона. Полученный результат представлен на рисунке 27.

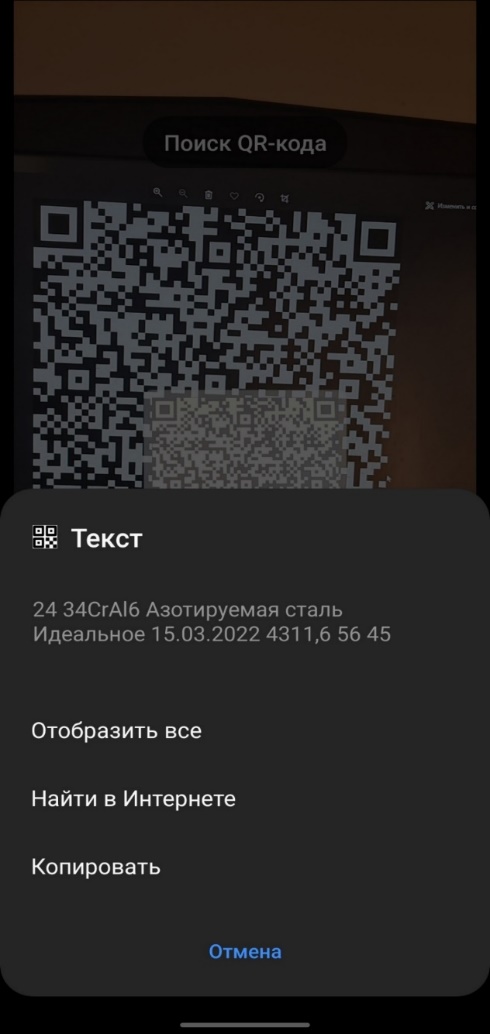


Рисунок 27 – Пример дешифровки QR-кода смартфоном

7) При нажатии на иконку с графиками открывается окно с отображением графика товарооборота предприятия и суммой реализованной продукции за период. Появившееся окно представлено на рисунке 28.

Здесь справа от верхнего графика находится компонент ListBox, в котором предлагается выбрать один из видов имеющейся на складе продукции. По нажатию кнопки «Отобразить» вырисовывается график товарооборота, который показывает сумму реализованной продукции одного вида товара по дням независимо от состояния. Сумма высчитывается путем сложения произведений количества на себестоимость каждого состояния одного вида товара. Этот график позволяет оценить рентабельность производства или закупки определенного вида товара.

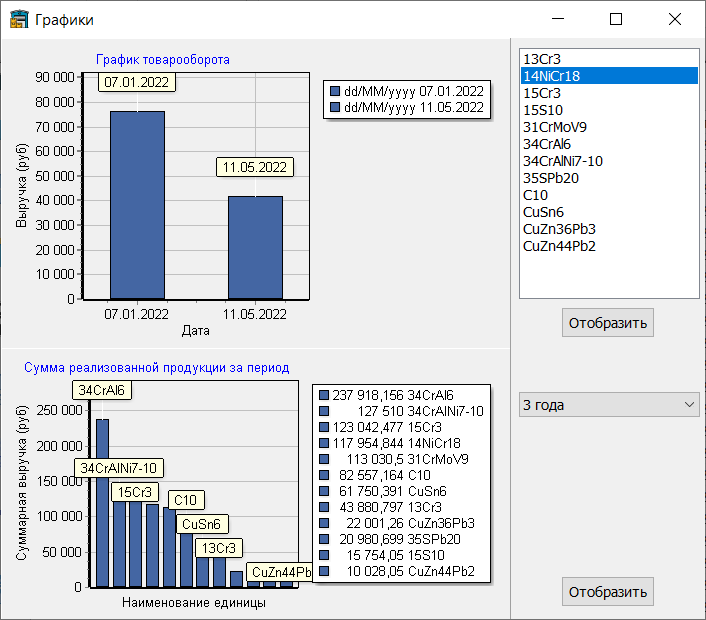


Рисунок 28 – Окно графиков СУСП

Справа от второго графика ниже находится компонент ComboBox, в котором предлагается выбрать период из предложенных вариантов: неделя, 2 недели, месяц, 3 месяца, 6 месяцев, год, 2 года, 3 года, 4 года, 5 лет, 10 лет.

За указанный промежуток времени показывается общая сумма выручки каждого имеющегося вида товара от наибольшей к наименьшей. Отсюда наглядно видно какому из видов товаров характерен наибольший товарооборот. Так как от товарооборота напрямую зависит расположение хранения продукции на складе, этот график представляет особый интерес для логистики складского помещения, а следовательно, способствует повышению эффективности использования пространства и повышению прибыли. Полученное окно показано на рисунке 29.

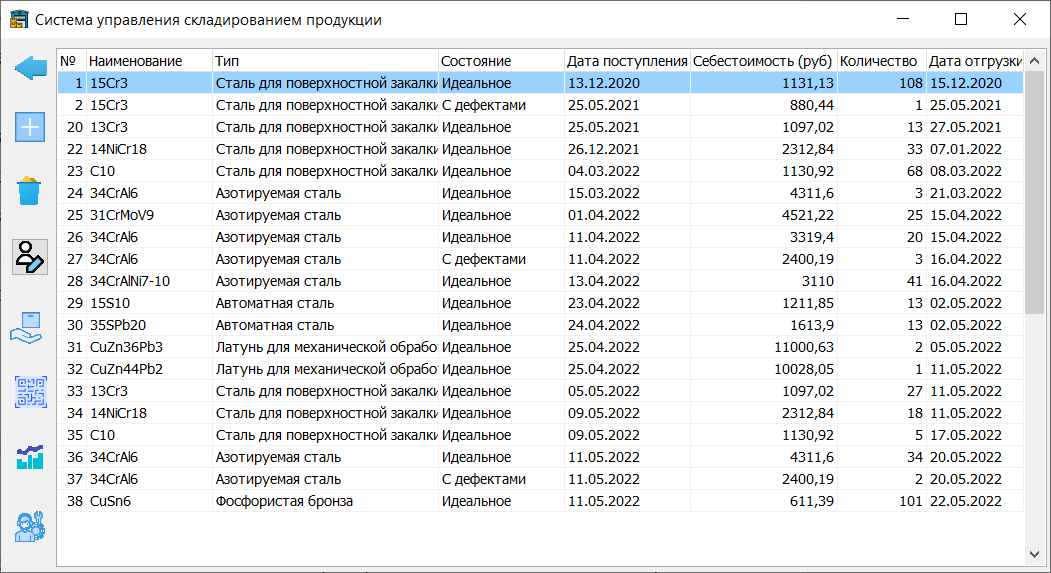


Рисунок 29 – Окно учета продукции в режиме администратора

## 3.2 Тестирование СУСП

Перед использованием СУСП в реальных условиях важно оценить устойчивость и быстроту работы основных вычислительно-ёмких участков программы.

При загрузке обеих таблиц БД свыше 500 полями видов продукции не было замечено изменений в работе программы. Поскольку в реальных условиях маловероятно использование программы с большим количеством полей, на этом результате проверка БД закончилась и было решено протестировать алгоритм генерации QR-кодов. Очевидно, что скорость работы алгоритма будет варьироваться в зависимости от мощности компьютера, однако было проведено несколько контрольных тестов для примерной оценки быстродействия без математики и оценки сложности алгоритма. Проверка включала в себя 6 запусков генерации кодов для разного количества единиц продукции. Результаты представлены в таблице 8.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Число товаров | 10 | 100 | 1000 | 2000 | 5000 | 10000 |
| Время (секунды) | Меньше 1 | Меньше 1 | 33 | 76 | 192 | 387 |

Таблица 8 – Результаты проверки скорости генерации QR-кодов

В результате видно, что с партиями порядка 100 единиц, алгоритм справляется мгновенно, а в случае, когда количество товаров около 10000 единиц, пришлось ждать 6,5 минут. Вероятнее всего, есть возможность ускорить эту операцию, однако программа рассчитана на маленькое производство, где товары поступают небольшими партиями и не слишком часто. В таком случае, СУСП справляется отлично. Тем не менее, из таблицы 8 также видно, что зависимость времени на вычисления от количества генерируемых файлов с кодами линейна. По грубой оценке, скорость кодировки на домашнем компьютере равна 27 файлов в секунду. При необходимости и наличии времени, можно кодировать и партии свыше 10000 единиц товаров.

Не менее важно учитывать получаемый размер каждого .bmp файла. Размер каждого полученного QR-кода напрямую зависит от кодируемой в него информации и выбранного уровня коррекции ошибок. При уровне коррекции ошибок H, в среднем файлы занимают 2 – 2.5 мегабайта памяти. В случае работы с партиями около 10000 единиц товаров важно учесть, чтобы в распоряжении имелось не меньше 25 гигабайт свободной памяти, что довольно много. Именно поэтому следует использовать уровень коррекции ошибок M, благодаря которому занимаемое количество памяти одним файлом с QR-кодом уменьшается более чем в 1,5 раза. Уровень коррекции ошибок L не рекомендуется использовать, так как в условиях реального производства риск порчи маркировки товара куда выше, в следствие чего безвозвратно потеряется возможность прочитать информацию с кода.

В итоге, разработанная программа оказалась полностью функциональной, относительно устойчивой к нагрузкам и достаточно эффективной для своих задач.

**Выводы по разделу**

В разделе была представлена рабочая версия разработанной СУСП с описанием её функционала, структуры, проведен тест на быстродействие.

**4 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ВКР**

* 1. **Введение**

Доля малого бизнеса в совокупном обороте малых, средних и крупных

предприятий в России составила 11,5%, что является минимумом с 2008 г. Также малые предприятия являются наиболее уязвимыми, и зачастую удержаться на рынке для них сложнее всего, особенно в кризисные ситуации. Практически любое производство подразумевает использование склада, однако для ведения учета продукции и организации складских помещений требуется соответствующая программа. Большинство подобных разработок являются платными и ориентированы на коммерческие компании, с избытком функционала для небольшого цеха или предприятия. Именно поэтому было решено разработать полностью бесплатный аналог, более подходящий для малых предприятий.

В результате выполнения ВКР была разработана система управления складированием продукции. При написании работы была использована среда разработки Embarcadero C++ Builder, задействована библиотека libqrencode и использованы компоненты ADO для работы с базами данных.

* 1. **Составление детализированного плана работ**

Для расчета затрат на этапе проектирования необходимо определить продолжительность каждой работы. Под проектированием понимается совокупность работ, которые необходимо выполнить. За основу для расчета был взят бакалаврский календарный план. Продолжительность работ осуществляется расчетным путем с помощью экспертных оценок по формуле

,

где  - ожидаемая длительность j-й работы;   
и - наименьшая и наибольшая, по мнению эксперта, длительность работы.

Перечень работ по разработке системы и трудоемкость их выполнения приведены в таблице 9

Таблица 9 – Трудоемкость работ по разработке СУСП

|  |  |  |
| --- | --- | --- |
| Наименование работы | Длительность работы, чел./дни | |
| Руководитель | Исполнитель |
| Разработка технического задания | 1 | 1 |
| Поиск необходимой информации | - | 5 |
| Обработка и анализ прочитанной литературы | - | 7 |
| Разработка технического задания | 3 | 6 |
| Выбор и подготовка к работе среды программирования | - | 1 |
| Разработка программы | 5 | 20 |
| Моделирование | 10 | 15 |
| Отладка и тестирование программы | 1 | 3 |
| Написание пояснительной записки | 1 | 1 |
| Итого | 21 | 59 |

## 4.3 Основная заработная плата

Далее необходимо определить ставку заработной платы за единицу времени каждого исполнителя. Ставка заработной платы за единицу времени определяется исходя из месячной заработной платы соответствующего исполнителя. Для определения дневной ставки заработной платы необходимо разделить месячный оклад на количество рабочих дней в месяце (21 рабочий день). Для студента в качестве месячной заработной платы принимается заработная плата инженера.

Заработная плата инженера принимается за 25200 р.



Заработная плата руководителя - 57500р.



Расходы на основную заработную плату исполнителей определяются по формуле:

, (5.3)

где – расходы на основную заработную плату руб.; k – количество исполнителей; – время, затраченное i–м исполнителем на исследования, дни или часы; – ставка i–го исполнителя руб./день или руб./час.

Подставив исходные данные в формулу для расчета основной заработной платы, получилась сумма размером .

## 4.4 Дополнительная заработная плата

Данную дополнительную заработную плату можно определить по формуле:

,

где расходы на дополнительную заработную плату исполнителей руб.; расходы на основную заработную плату исполнителей руб.; норматив дополнительной заработной платы %.

Принимаем норматив дополнительной заработной платы при работе над ВКР равным 14%.

Отсюда:



Исходя из расчетов, дополнительная заработная плата будет равна руб.

## 4.5 Отчисления на социальные нужды

Отчисления из основной и дополнительной заработной платы на страховое взносы, связанные с обязательным пенсионным, социальным и медицинским страхованием, вычисляется по следующей формуле:

,

где отчисления на социальные нужды с заработной платы руб.; расходы на основную заработную плату исполнителей руб.; расходы на дополнительную заработную плату исполнителей руб.; норматив отчислений на страховые взносы на обязательное социальное, пенсионное и медицинское страхование (%), которое на 20.05.2022 составляет 30%.

Получается:

.

Итого отчисления из основной и дополнительной заработной платы составили рублей.

## 4.6 Сырье и материалы

Была проведена оценка расходов на материалы и комплектующие, необходимые для выполнения работы, относящиеся к прямым издержкам и включенные непосредственно в себестоимость разработанного образца.

Цена расходных материалов была определена из учета цен, действующих в розничных магазинах.

Затраты на покупные материалы определяются по формуле:

,

где – затраты на покупные материалы руб.;– количество *i*-тых материалов входящих в единицу продукции штук;– цена приобретения единицы *i*-го материала рублей/штук; –норма транспортно-заготовительных расходов %.

Данные покупных материалов представлены в таблице 10.

Таблица 10 – Затраты на покупные материалы

|  |  |  |  |
| --- | --- | --- | --- |
| Материал | Норма рас­хода на ед. продукции | Цена единицы, руб./шт. | Сумма на единицу продукции, руб. |
| Ручка шариковая, шт. | 1 | 35 | 35 |
| Бумага для оргтехники, пачка | 1 | 350 | 350 |
| USB-накопитель | 1 | 670 | 670 |

Затраты на материалы определяются по следующей формуле:

|  |  |
| --- | --- |
| , |  |

где  – затраты на материалы, руб.;  – индекс вида материала;  – норма расхода -того материала на единицу продукции, ед.;  – цена приобретения единицы -того материала, руб./ед.;  – норма транспортно-заготовительных расходов, принимается равной 10 %.

Подставив значения в формулу выше, получается:



## 4.7 Расчет затрат на эксплуатацию оборудования и его содержание

Затраты на эксплуатацию оборудования и его содержание определяются из расчета на 1 день работы оборудования с учетом его производительности и стоимости. Формула для определения этих затрат выглядит следующим образом:

|  |
| --- |
| , |

где  – затраты на содержание и эксплуатацию оборудования, руб.;  – расчётная себестоимость одного машино-дня работы оборудования на -й технологической операции составляет 240 руб./м-д;  – количество машино-дней, затрачиваемых на выполнение -й технологической операции, м-д.

Подставив значения в формулу выше, при условии, что стоимость м-д для каждой технологической операции одинакова, и количество затрачиваемых исполнителем и руководителем дней равно 80, получается: .

## 4.8 Расчёт амортизационных отчислений

Амортизационные отчисления рассчитываются по следующей формуле:

|  |  |
| --- | --- |
| , |  |

где  – амортизационные отчисления за год по -му основному средству, руб.;  – первоначальная стоимость -го основного средства, руб.;  – годовая норма амортизации -го основного средства (составляет 50 %). Стоимость ноутбука Lenovo Legion – .

Подставив стоимость в формулу выше, получается:



Величина амортизационных отчислений по -му основному средству, используемому студентом при работе над ВКР, определяется по формуле:

|  |  |
| --- | --- |
| , |  |

где  – амортизационные отчисления по -му основному средству, используемому студентом при работе над ВКР, руб.;  – время использования студентом -го основного средства, дни;  – количество рабочих дней в году (247 дней).

Подставив значения в формулу выше, получается:



## 4.9 Калькуляция затрат на ВКР

Последним делом необходимо посчитать накладные расходы, которые составляют 50% от основной и дополнительной заработной платы:

руб.

Совокупная величина затрат, связанных с разработкой СУСП, представлена в таблице 11.

Таблица 11 – Затраты на ВКР

|  |  |
| --- | --- |
| Наименование статьи | Сумма, руб |
| Материалы | 1122 |
| Основная заработная плата | 128550 |
| Дополнительная заработная плата | 17997 |
| Отчисления на социальные нужды | 43964,1 |
| Накладные расходы | 73273,5 |
| Затраты на эксплуатацию и содержание оборудования | 19200 |
| Амортизационные отчисления | 6568,8 |
| ИТОГО | 290675,4 |

В итоге общая сумма затрат на разработку СУСП составила 290675,4 рублей. Наибольшую часть расходов составила основная заработная плата. Это связано с тем, что разработка программы почти не требует большого количества материальных затрат, однако занимает большое количество времени, потребляя умственный труд. Если учесть, что рабочий компьютер и основные материалы уже имелись в распоряжении, а специалисты достаточно квалифицированные и более трудолюбивые, то затраты должны сократиться на порядок. Тем не менее, даже несмотря на это, сэкономленные в перспективе средства при использовании СУСП однозначно превысят полученные затраты.

**Выводы по разделу**

В разделе были посчитаны суммарные затраты на ВКР, оценена экономическая целесообразность разработки СУСП, была произведена оценка наибольших расходов, было объяснено почему так происходит.

# ЗАКЛЮЧЕНИЕ

В рамках ВКР была разработана система управления складированием продукции. Программа успешно прошла проверку и выдала неплохие для своего уровня показатели быстродействия и устойчивости к перегрузке БД. СУСП получила очень простой, удобный и интуитивно понятный интерфейс, позволяющий практически мгновенно освоить программу любому человеку. Отсутствие необходимости платить за нее дает возможность получить разработке широкое распространение.

СУСП позволяет сократить количество работников склада предприятия, либо затрачиваемое ими время на опознавание продукции и её характеристик благодаря имеющемуся рабочему генератору QR-кодов, удобно и быстро управлять, искать или сортировать имеющуюся на складе продукцию, наглядно представлять масштабы товарооборота различных видов продукции, а также наблюдать во времени доход от выбранного товара.

Безусловно, СУСП еще не совершенна, однако наличие обратной связи с реальным её пользователем позволит в разы ускорить разработку и доработку программы.

В перспективе следует уделить внимание совершенствованию алгоритма генерации QR-кодов, расширить функционал программы, добавив, например, возможность считывания QR-кодов, а также сделать программу более защищенной.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Вергиев Н.С. 9 принципов складского хозяйства // Современный склад. 2002, вып. 1. С. 8-9.

2. Вегст К., Вегст. М. Ключ к сталям. СПб.: Изд-во Профессия, 2006.

21 с

3. Шевченко А.В. Организация баз данных в интегрированных системах управления процессами производства: учеб. пособие. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2014. 13 с

4. Лекция 2: ADO. Связь с таблицей MS Access. ИНТУИТ – Национальный открытый университет. Режим доступа: <https://intuit.ru/studies/curriculums/18833/courses/470/lecture/20867?page=1> (Дата обращения: 18.05.2022).

5. Жанкоразова Н.Н., Абилдаева Г. Б., Жанадил О. Технология ADO и средства доступа к реляционным базам данных // Молодой ученый. — 2015. — № 11 (91). — С. 156-158.

6. Парфенов Ю.П., Иноземцев А.В. / Разработка приложений для обработки баз данных на основе технологии ADO. Екатеринбург: Издательство УМЦ УПИ, 2003, 15 с.

7. Как работает QR-код?. Хабр. Режим доступа:  [https://habr.com/ru/company/skillfactory/blog/528320/](%20https://habr.com/ru/company/skillfactory/blog/528320/) (Дата обращения: 21.05.2022).

8. Algorithm for generating a QR code. Weekly-Geekly. Режим доступа: <https://weekly-geekly.imtqy.com/articles/172525/index.html>(Дата обращения: 21.05.2022).

9. Код Рида - Соломона. ВикипедиЯ. Свободная энциклопедия. Режим доступа: <https://ru.wikipedia.org/wiki/Код_Рида_—_Соломона> (Дата обращения: 21.05.2021).

# ПРИЛОЖЕНИЕ А

Код программы:

void \_\_fastcall Tform\_main::Button1Click(TObject \*Sender)

{

const char\* szSourceString = QRCODE\_TEXT;

unsigned int unWidth, x, y, l, n, unWidthAdjusted, unDataBytes;

unsigned char\* pRGBData, \*pSourceData, \*pDestData;

QRcode\* pQRC;

FILE\* f;

/\*

\* Параметры:

\* string: входная строка; должна завершаться нулем

\* version: версия символа; если 0, библиотека выбирает минимальную версию входных данных

\* level: уровень коррекции ошибок

\* hint: указывает библиотеке на то, как должны кодироваться не алфавитные символы

\* QR\_MODE\_KANJI - иероглифы будут кодироваться как Shif-JIS

\* QR\_MODE\_8 - все не алфавитные символы будут кодироваться как есть.

\* casesensitive: case-sensitive(1) или нет(0).

\* return: экземпляр класса QRcode; при ошибке возвращается NULL и errno устанавливает ошибку

\* EINVAL: некорректный входной объект

\* ENOMEM: не удается выделить память для входных объектов

\* ERANGE: входные данные слишком большие

\*/

// Вычисление QRcode

if ((pQRC = QRcode\_encodeString(szSourceString, 0, QR\_ECLEVEL\_H, QR\_MODE\_8, 1)))

{

unWidth = pQRC->width;

unWidthAdjusted = unWidth \* OUT\_FILE\_PIXEL\_PRESCALER \* 3;

if (unWidthAdjusted % 4)

unWidthAdjusted = (unWidthAdjusted / 4 + 1) \* 4;

unDataBytes = unWidthAdjusted \* unWidth \* OUT\_FILE\_PIXEL\_PRESCALER;

if (!(pRGBData = (unsigned char\*)malloc(unDataBytes)))

{

exit(1);

}

memset(pRGBData, 0xff, unDataBytes);

// Подготовка заголовков bmp

BITMAPFILEHEADER kFileHeader;

kFileHeader.bfType = 0x4d42; // "BM"

kFileHeader.bfSize = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER) + unDataBytes;

kFileHeader.bfReserved1 = 0;

kFileHeader.bfReserved2 = 0;

kFileHeader.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER);

BITMAPINFOHEADER kInfoHeader;

kInfoHeader.biSize = sizeof(BITMAPINFOHEADER);

kInfoHeader.biWidth = unWidth \* OUT\_FILE\_PIXEL\_PRESCALER;

kInfoHeader.biHeight = -((int)unWidth \* OUT\_FILE\_PIXEL\_PRESCALER); kInfoHeader.biPlanes = 1;

kInfoHeader.biBitCount = 24;

kInfoHeader.biCompression = BI\_RGB;

kInfoHeader.biSizeImage = 0;

kInfoHeader.biXPelsPerMeter = 0;

kInfoHeader.biYPelsPerMeter = 0;

kInfoHeader.biClrUsed = 0;

kInfoHeader.biClrImportant = 0;

// Конвертирование битов QrCode в bmp пиксели

pSourceData = pQRC->data;

for(y = 0; y < unWidth; y++)

{

pDestData = pRGBData + unWidthAdjusted \* y \* OUT\_FILE\_PIXEL\_PRESCALER; for(x = 0; x < unWidth; x++)

{

if (\*pSourceData & 1)

for(l = 0; l < OUT\_FILE\_PIXEL\_PRESCALER; l++)

for(n = 0; n < OUT\_FILE\_PIXEL\_PRESCALER; n++)

{

\*(pDestData + n \* 3 + unWidthAdjusted \* l) = PIXEL\_COLOR\_B;

\*(pDestData + 1 + n \* 3 + unWidthAdjusted \* l) = PIXEL\_COLOR\_G;

\*(pDestData + 2 + n \* 3 + unWidthAdjusted \* l) = PIXEL\_COLOR\_R;

}

pDestData += 3 \* OUT\_FILE\_PIXEL\_PRESCALER;

pSourceData++;

}

}

if (!(fopen\_s(&f, OUT\_FILE, "wb")))

{

fwrite(&kFileHeader, sizeof(BITMAPFILEHEADER), 1, f);

fwrite(&kInfoHeader, sizeof(BITMAPINFOHEADER), 1, f);

fwrite(pRGBData, sizeof(unsigned char), unDataBytes, f);

fclose(f);

}

else

{

//unable to open file

exit(1);

}

free(pRGBData);

QRcode\_free(pQRC);

}

else

{

// NULL returned

exit(1);

}

}

void \_\_fastcall Tform\_add\_storage::bt\_ok\_storageClick(TObject \*Sender)

{

if (edit\_add\_storage\_name->Text == ""

|| edit\_add\_storage\_type->Text == ""

|| edit\_add\_storage\_condition->Text == ""

|| edit\_add\_storage\_date->Text == ""

|| edit\_add\_storage\_price->Text == ""

|| edit\_add\_storage\_count->Text == "")

{

ShowMessage("Все поля должны быть заполнены");

}

else if (StrToFloat(edit\_add\_storage\_price->Text) <= 0 || edit\_add\_storage\_count->Text.ToInt() <= 0)

{

ShowMessage("Числа должны быть неотрицательными");

}

else {

if (form\_storage->DBGridStorage->Visible) {

std::vector<AnsiString> edit\_mas = {edit\_add\_storage\_name->Text, edit\_add\_storage\_type->Text,

edit\_add\_storage\_condition->Text, edit\_add\_storage\_date->Text,

edit\_add\_storage\_price->Text, edit\_add\_storage\_count->Text};

for (int i = 1; i < form\_storage->ADOTableStorage->RecordCount+1; ++i) {

int k = 0;

form\_storage->DBGridStorage->DataSource->DataSet->RecNo = i;

for (int j = 1; j<6; ++j){

if (edit\_mas[j-1] != form\_storage->ADOTableStorage->Fields->Fields[j]->AsAnsiString) break;

k++;

}

if (k == 5) {

form\_storage->ADOTableStorage->Edit();

form\_storage->ADOTableStorage->Fields->Fields[6]->AsInteger += std::atoi(edit\_mas[5].c\_str());

form\_storage->ADOTableStorage->Post();

ShowMessage("Было добавлено " + edit\_mas[5] + " единиц(ы) продукции.");

break;

}

else if (i == form\_storage->ADOTableStorage->RecordCount) {

form\_storage->ADOTableStorage->Insert();

form\_storage->ADOTableStorage->FieldByName("Наименование")->Value = edit\_add\_storage\_name->Text;

form\_storage->ADOTableStorage->FieldByName("Тип")->Value = edit\_add\_storage\_type->Text;

form\_storage->ADOTableStorage->FieldByName("Состояние")->Value = edit\_add\_storage\_condition->Text;

form\_storage->ADOTableStorage->FieldByName("Дата поступления")->Value = edit\_add\_storage\_date->Text;

form\_storage->ADOTableStorage->FieldByName("Себестоимость (руб)")->Value = edit\_add\_storage\_price->Text;

form\_storage->ADOTableStorage->FieldByName("Количество")->Value = edit\_add\_storage\_count->Text;

form\_storage->ADOTableStorage->Post();

form\_add\_storage->Close();

break;

}

}

}

else{

std::vector<AnsiString> edit\_mas = {edit\_add\_storage\_name->Text, edit\_add\_storage\_type->Text, edit\_add\_storage\_condition->Text, edit\_add\_storage\_date->Text, edit\_add\_storage\_price->Text, edit\_add\_storage\_count->Text, edit\_add\_storage\_date2->Text};

for (int i = 1; i < form\_storage->ADOTableSold->RecordCount+1; ++i) {

int k = 0;

form\_storage->DBGridSold->DataSource->DataSet->RecNo = i;

for (int j = 1; j<8; ++j){

if (j != 6 && edit\_mas[j-1] != form\_storage->ADOTableSold->Fields->Fields[j]->AsAnsiString) break;

k++;

}

if (k == 7) {

form\_storage->ADOTableSold->Edit();

form\_storage->ADOTableSold->Fields->Fields[6]->AsInteger += std::atoi(edit\_mas[5].c\_str());

form\_storage->ADOTableSold->Post();

ShowMessage("Было добавлено " + edit\_mas[5] + " единиц(ы) продукции.");

break;

}

else if (i == form\_storage->ADOTableSold->RecordCount) {

form\_storage->ADOTableSold->Insert();

form\_storage->ADOTableSold->FieldByName("Наименование")->Value = edit\_add\_storage\_name->Text;

form\_storage->ADOTableSold->FieldByName("Тип")->Value = edit\_add\_storage\_type->Text;

form\_storage->ADOTableSold->FieldByName("Состояние")->Value = edit\_add\_storage\_condition->Text;

form\_storage->ADOTableSold->FieldByName("Дата поступления")->Value = edit\_add\_storage\_date->Text;

form\_storage->ADOTableSold->FieldByName("Себестоимость (руб)")->Value = edit\_add\_storage\_price->Text;

form\_storage->ADOTableSold->FieldByName("Количество")->Value = edit\_add\_storage\_count->Text;

form\_storage->ADOTableSold->FieldByName("Дата отгрузки")->Value = edit\_add\_storage\_date2->Text;

form\_storage->ADOTableSold->Post();

form\_add\_storage->Close();

break;

}

}

}

form\_storage->ADOTableSold->Active = False;

form\_storage->ADOTableSold->Active = True;

}

}

\_\_fastcall Tform\_graphics::Tform\_graphics(TComponent\* Owner)

: TForm(Owner)

{

TDateTime temp\_date = 30/12/1899;

long int temp\_N = 0;

float temp\_price = 0;

//-----------------------------------

form\_storage->ADOTableSold->IndexFieldNames = "№";

form\_storage->ADOTableSold->First();

for (long int i = 0; i < form\_storage->ADOTableSold->RecordCount; ++i, form\_storage->ADOTableSold->Next() ) {

ListBoxGraphics->Items->Add(form\_storage->ADOTableSold->Fields->Fields[1]->AsAnsiString);

if (i && form\_storage->ADOTableSold->Fields->Fields[0]->AsLargeInt == temp\_N) {

ListBoxGraphics->Items->Delete(i);

if (form\_storage->ADOTableSold->Fields->Fields[7]->AsDateTime == temp\_date)

{

temp\_price = sold\_map.find(form\_storage->ADOTableSold->Fields->Fields[1]->AsAnsiString)->second.first;

sold\_map.erase(sold\_map.find(form\_storage->ADOTableSold->Fields->Fields[1]->AsAnsiString) ) ;

sold\_map.insert(std::pair < AnsiString, std::pair < float, TDateTime > > (form\_storage->ADOTableSold->Fields->Fields[1]->AsAnsiString, {form\_storage->ADOTableSold->Fields->Fields[5]->AsFloat \* form\_storage->ADOTableSold->Fields->Fields[6]->AsFloat + temp\_price, form\_storage->ADOTableSold->Fields->Fields[7]->AsDateTime}) );

}

}

temp\_date = form\_storage->ADOTableSold->Fields->Fields[7]->AsDateTime;

temp\_N = form\_storage->ADOTableSold->Fields->Fields[0]->AsLargeInt;

sold\_map.insert(std::pair < AnsiString, std::pair < float, TDateTime > > (form\_storage->ADOTableSold->Fields->Fields[1]->AsAnsiString, {form\_storage->ADOTableSold->Fields->Fields[5]->AsFloat \* form\_storage->ADOTableSold->Fields->Fields[6]->AsFloat, form\_storage->ADOTableSold->Fields->Fields[7]->AsDateTime}) );

}

for (auto i = 0; i < ListBoxGraphics->Count; ++i) {

if ( i && ListBoxGraphics->Items->Strings[i] == ListBoxGraphics->Items->Strings[i-1]) {

ListBoxGraphics->Items->Delete(i-1);

i = 0;

}

}

}

void \_\_fastcall Tform\_graphics::bt\_graphicsClick(TObject \*Sender)

{

//ChartGraphics->RemoveAllSeries();

//ChartGraphics->SeriesList->Clear();

Series1->Clear();

if(!(ListBoxGraphics->ItemIndex == -1)){

std::multimap <AnsiString, std::pair <float, TDateTime> > sold\_map2(sold\_map);

while (sold\_map2.find(ListBoxGraphics->Items->operator [](ListBoxGraphics->ItemIndex)) != sold\_map2.end() ){

Series1->Add(sold\_map2.find(ListBoxGraphics->Items->operator [](ListBoxGraphics->ItemIndex))->second.first,

sold\_map2.find(ListBoxGraphics->Items->operator [](ListBoxGraphics->ItemIndex))->second.second);

//ShowMessage(sold\_map.find(ListBoxGraphics->Items->operator [](ListBoxGraphics->ItemIndex))->second.second);

sold\_map2.erase(sold\_map2.find(ListBoxGraphics->Items->operator [](ListBoxGraphics->ItemIndex)));

}

}

}

void \_\_fastcall Tform\_graphics::bt\_sold\_graphicsClick(TObject \*Sender)

{

Series2->Clear();

double period;

switch (ComboBoxGraphics->ItemIndex) {

case 0: {

period = 7;

break;

}

case 1: {

period = 14;

break;

}

case 2: {

period = 30.4375;

break;

}

case 3: {

period = 91.3125;

break;

}

case 4: {

period = 182.625;

break;

}

case 5: {

period = 365.25;

break;

}

case 6: {

period = 730.5;

break;

}

case 7: {

period = 1095.75;

break;

}

case 8: {

period = 1461;

break;

}

case 9: {

period = 1826.25;

break;

}

case 10: {

period = 3652.5;

break;

}

}

std::multimap <AnsiString, std::pair <float, TDateTime> > sold\_map2(sold\_map);

for (auto it = sold\_map2.begin(); it != sold\_map.end(); it = sold\_map2.begin()) {

if (it->second.second >= (Date() - period) && it->second.second <= Date() ) {

float temp = 0;

AnsiString temp\_name = it->first;

while (sold\_map2.find(temp\_name) != sold\_map2.end()){

//Series2->Add(sold\_map2.find(it->first)->second.first, sold\_map2.find(it->first)->second.second);

temp += sold\_map2.find(temp\_name)->second.first;

temp\_name = sold\_map2.find(temp\_name)->first;

sold\_map2.erase(sold\_map2.find(temp\_name));

}

Series2->Add(temp, temp\_name);

//sold\_map2.erase(sold\_map2.find(it->first));

}

else if (sold\_map2.empty()) break;

else sold\_map2.erase(it);

}

Series2->XValues->Order = loNone;

Series2->YValues->Order = loDescending;

Series2->YValues->Sort();

Series2->XValues->FillSequence();

Series2->Repaint();

}

void \_\_fastcall Tform\_sell::bt\_sell\_okClick(TObject \*Sender)

{

if (edit\_sell->Text.ToInt() <= 0 ) {

ShowMessage("Вы должны ввести положительное число");

}

else if (edit\_sell->Text.ToInt() > form\_storage->ADOTableStorage->Fields->Fields[6]->AsInteger) {

ShowMessage("В базе нет столько единиц продукции");

}

else {

if (form\_storage->DBGridStorage->Visible) {

bool flag = 1;

form\_storage->ADOTableStorage->Edit();

form\_storage->ADOTableStorage->Fields->Fields[6]->AsInteger -= edit\_sell->Text.ToInt();

form\_storage->ADOTableStorage->Post();

// Заполнение SOLD из STORAGE

form\_storage->ADOTableSold->IndexFieldNames = "№";

TLocateOptions loCaseInsensitive;

loCaseInsensitive.Clear();

form\_storage->ADOTableSold->Locate("№", form\_storage->ADOTableStorage->Fields->Fields[0]->AsLargeInt, loCaseInsensitive);

for (form\_storage->ADOTableSold->RecNo; form\_storage->ADOTableSold->Fields->Fields[0]->AsLargeInt == form\_storage->ADOTableStorage->Fields->Fields[0]->AsLargeInt; form\_storage->ADOTableSold->RecNo++)

{

if (form\_storage->ADOTableSold->Fields->Fields[1]->AsAnsiString == form\_storage->ADOTableStorage->Fields->Fields[1]->AsAnsiString

&& form\_storage->ADOTableSold->Fields->Fields[2]->AsAnsiString == form\_storage->ADOTableStorage->Fields->Fields[2]->AsAnsiString

&& form\_storage->ADOTableSold->Fields->Fields[3]->AsAnsiString == form\_storage->ADOTableStorage->Fields->Fields[3]->AsAnsiString

&& form\_storage->ADOTableSold->Fields->Fields[4]->AsDateTime == form\_storage->ADOTableStorage->Fields->Fields[4]->AsDateTime

&& form\_storage->ADOTableSold->Fields->Fields[5]->AsFloat == form\_storage->ADOTableStorage->Fields->Fields[5]->AsFloat

&& form\_storage->ADOTableSold->Fields->Fields[7]->AsDateTime == Date() )

{

form\_storage->ADOTableSold->Edit();

form\_storage->ADOTableSold->Fields->Fields[6]->AsLargeInt += StrToInt64(edit\_sell->Text);

form\_storage->ADOTableSold->Post();

flag = !flag;

break;

}

}

if (flag) {

form\_storage->ADOTableSold->Last();

form\_storage->ADOTableSold->Insert();

form\_storage->ADOTableSold->Fields->Fields[0]->AsLargeInt = form\_storage->ADOTableStorage->Fields->Fields[0]->AsLargeInt;

form\_storage->ADOTableSold->Fields->Fields[1]->AsAnsiString = form\_storage->ADOTableStorage->Fields->Fields[1]->AsAnsiString;

form\_storage->ADOTableSold->Fields->Fields[2]->AsAnsiString = form\_storage->ADOTableStorage->Fields->Fields[2]->AsAnsiString;

form\_storage->ADOTableSold->Fields->Fields[3]->AsAnsiString = form\_storage->ADOTableStorage->Fields->Fields[3]->AsAnsiString;

form\_storage->ADOTableSold->Fields->Fields[4]->AsDateTime = form\_storage->ADOTableStorage->Fields->Fields[4]->AsDateTime;

form\_storage->ADOTableSold->Fields->Fields[5]->AsFloat = form\_storage->ADOTableStorage->Fields->Fields[5]->AsFloat;

form\_storage->ADOTableSold->Fields->Fields[6]->AsLargeInt = StrToInt64(edit\_sell->Text);

form\_storage->ADOTableSold->Fields->Fields[7]->AsDateTime = Date();

form\_storage->ADOTableSold->Post();

form\_sell->Close();

}

// GRID SOLD STRUCT

/\* sell\_grid sell\_grid\_temp {form\_storage->ADOTableStorage->Fields->Fields[0]->AsLargeInt,

form\_storage->ADOTableStorage->Fields->Fields[1]->AsAnsiString,

form\_storage->ADOTableStorage->Fields->Fields[2]->AsAnsiString,

form\_storage->ADOTableStorage->Fields->Fields[3]->AsAnsiString,

form\_storage->ADOTableStorage->Fields->Fields[4]->AsDateTime,

form\_storage->ADOTableStorage->Fields->Fields[5]->AsLargeInt,

StrToFloat(edit\_sell->Text),

Date()

};

\*/

form\_sell->Close();

}

else {

bool flag = 1;

form\_storage->ADOTableSold->Edit();

form\_storage->ADOTableSold->Fields->Fields[6]->AsInteger -= edit\_sell->Text.ToInt();

form\_storage->ADOTableSold->Post();

// Заполнение SOLD из STORAGE

form\_storage->ADOTableStorage->IndexFieldNames = "Наименование";

TLocateOptions loCaseInsensitive;

loCaseInsensitive.Clear();

form\_storage->ADOTableStorage->Locate("Наименование", form\_storage->ADOTableSold->Fields->Fields[1]->AsAnsiString, loCaseInsensitive);

for (form\_storage->ADOTableStorage->RecNo; form\_storage->ADOTableStorage->Fields->Fields[1]->AsAnsiString == form\_storage->ADOTableSold->Fields->Fields[1]->AsAnsiString; form\_storage->ADOTableStorage->RecNo++)

{

if (form\_storage->ADOTableStorage->Fields->Fields[2]->AsAnsiString == form\_storage->ADOTableSold->Fields->Fields[2]->AsAnsiString

&& form\_storage->ADOTableStorage->Fields->Fields[3]->AsAnsiString == form\_storage->ADOTableSold->Fields->Fields[3]->AsAnsiString

&& form\_storage->ADOTableStorage->Fields->Fields[4]->AsDateTime == form\_storage->ADOTableSold->Fields->Fields[4]->AsDateTime

&& form\_storage->ADOTableStorage->Fields->Fields[5]->AsFloat == form\_storage->ADOTableSold->Fields->Fields[5]->AsFloat)

{

form\_storage->ADOTableStorage->Edit();

form\_storage->ADOTableStorage->Fields->Fields[6]->AsLargeInt += StrToInt64(edit\_sell->Text);

form\_storage->ADOTableStorage->Post();

flag = !flag;

break;

}

}

if (flag) {

form\_storage->ADOTableStorage->Last();

form\_storage->ADOTableStorage->Insert();

//form\_storage->ADOTableStorage->Fields->Fields[0]->AsLargeInt = form\_storage->ADOTableSold->Fields->Fields[0]->AsLargeInt;

form\_storage->ADOTableStorage->Fields->Fields[1]->AsAnsiString = form\_storage->ADOTableSold->Fields->Fields[1]->AsAnsiString;

form\_storage->ADOTableStorage->Fields->Fields[2]->AsAnsiString = form\_storage->ADOTableSold->Fields->Fields[2]->AsAnsiString;

form\_storage->ADOTableStorage->Fields->Fields[3]->AsAnsiString = form\_storage->ADOTableSold->Fields->Fields[3]->AsAnsiString;

form\_storage->ADOTableStorage->Fields->Fields[4]->AsDateTime = form\_storage->ADOTableSold->Fields->Fields[4]->AsDateTime;

form\_storage->ADOTableStorage->Fields->Fields[5]->AsFloat = form\_storage->ADOTableSold->Fields->Fields[5]->AsFloat;

form\_storage->ADOTableStorage->Fields->Fields[6]->AsLargeInt = StrToInt64(edit\_sell->Text);

form\_storage->ADOTableStorage->Post();

form\_sell->Close();

}

form\_sell->Close();

}

}

}