МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Моделирование процесса работы системы “Накопитель окрашенных кузовов”

Обучающийся А.А. Анесян
(Инициалы И.О. Фамилия) (личная подпись)

Руководитель Доктор ф.-м. наук, профессор, А. И. Сафронов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант К.п.н., доцент, О.Н. Брега
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Тема выпускной квалификационной работы: Моделирование процесса работы системы ”Накопитель окрашенных кузовов”.

Целью выпускной квалификационной работы является разработка модели системы “Накопитель окрашенных кузовов”.

Объектом выпускной квалификационной работы является процесс работы системы “Накопитель окрашенных кузовов”.

Предмет исследования – построение модели системы “Накопитель окрашенных кузовов”.

В данной выпускной квалификационной работе исследуется построение систем “Накопитель окрашенных кузовов”. Рассматриваются основные виды моделирования систем и методы реализации систем.

Структура выпускной квалификационной работы состоит из введения, трех глав, заключения, списка используемой литературы и используемых источников.

Во введении описывается актуальность проводимого исследования, формулируется цель и ставятся задачи, которые необходимо решить.

В первой главе приводится информация об информационных системах и процессе их реновации.

Во второй главе рассматривается процесс выбора необходимого метода моделирования и математической модели.

В третьей главе рассматривается процесс программной реализации имитационной модели.

Бакалаврская работа состоит из 49с. пояснительной записки, включая 22 рисунка.

Abstract

The topic of the final qualifying work: Modeling the process of operation of the system ”Storage of painted bodies".

The purpose of the final qualifying work is to develop a model of the “Painted body Storage” system.

The object of the final qualification work is the process of operation of the system “Storage of painted bodies".

The subject of the study is the construction of a model of the “Painted body Storage” system.

In this final qualifying work, the construction of “Painted body Storage” systems is investigated. The main types of system modeling and methods of system implementation are considered.

The structure of the final qualifying work consists of an introduction, three chapters, a conclusion, a list of the literature used and the sources used.

The introduction describes the relevance of the research, formulates the goal and sets the tasks that need to be solved.

The first chapter provides information about information systems and the process of their renovation.

The second chapter discusses the process of choosing the necessary modeling method and mathematical model.

The third chapter discusses the process of software implementation of the simulation model.

The bachelor's thesis consists of 49 pages of explanatory notes, including 22 drawings.

Содержание

Введение 5

1 Реновация информационных систем 8

1.1 Информационные системы 8

1.2 Процесс реновации 10

2 Моделирование процесса работы системы 11

2.1 Моделирование процессов 11

2.2 Методы моделирования 13

2.3 Имитационное моделирование 15

2.4 Система массового обслуживания 17

2.5 Математическая модель 21

3 Проектирование и реализация имитационной модели 24

3.1 Обзор и выбор технологий программирования 24

3.1.1 Обзор и выбор языка программирования 24

3.1.2 Обзор и выбор среды разработки 26

3.2 Проектирование архитектуры программного обеспечения 29

3.3 Реализация программы 30

Заключение 44

Список используемой литературы и используемых источников 46

Введение

Система накопитель окрашенных кузовов является важным звеном в производственном процессе автомобильной промышленности. Её целью является складирование окрашенных кузовов для последующей сборки автомобилей. При этом, процесс работы системы должен быть максимально эффективным и безопасным.

С развитием технологий и изменением бизнес-процессов, информационные системы становятся все более устаревшими и неэффективными. В этой связи, предприятия вынуждены проводить процесс реновации информационных систем, который направлен на улучшение их производительности, снижение затрат и повышение качества обслуживания клиентов.

Однако, процесс реновации информационных систем является сложным и многогранным процессом, который может включать в себя изменения в архитектуре системы, замену оборудования, перенос данных и т.д. При неправильном подходе, процесс реновации может привести к сбоям в работе системы и негативным последствиям для бизнеса. Поэтому для обеспечения надёжной и налаженной работы необходимо составить точную модель системы, на основе которой и будет реализован конечный продукт.

Для оптимизации процесса работы системы накопителя окрашенных кузовов, используется моделирование процессов. Моделирование процессов является процессом создания математических моделей, которые могут анализировать и предсказывать поведение системы в определенных условиях.

При моделировании процесса работы системы накопителя окрашенных кузовов, важно учитывать такие факторы, как количество кузовов, скорость их движения, взаимодействие с другими системами производства и др. Для этого используются различные методы и технологии моделирования, такие как аналитическое моделирование, эмпирическое моделирование, статистическое моделирование, а также имитационное моделирование.

Для достижения поставленных целей наиболее подходящим вариантом является имитационное моделирование.

Одним из основных преимуществ имитационного моделирования процесса работы системы накопителя окрашенных кузовов является возможность оптимизации работы системы без необходимости проведения реальных испытаний и экспериментов. Модель позволяет исследовать различные варианты работы системы, оценить их эффективность и принять решение об оптимальном варианте.

Цель данного проекта – разработать модель процесса работы системы, которая позволит её проанализировать и на её основе реализовать будущую систему.

В соответствии с поставленной целью в работе определены следующие задачи:

- проанализировать предметную область;
- рассмотреть методы моделирования и выбрать наиболее подходящий;
- составить математическую модель системы;
- рассмотреть технологии программирования и выбрать наиболее подходящие;
- спроектировать архитектуру программного обеспечения;
- реализовать программное обеспечение;
- протестировать разработанную программу.

Бакалаврская работа состоит из введения, трех разделов, заключения и списка литературы.

Во введении описывается актуальность данной задачи, формулируется цель и ставятся задачи, поставленные в работе.

В первом разделе описывается процесс реновации информационных систем.

Во втором разделе описываются методы моделирования, выбирается наиболее подходящий, а также составляется математическая модель.

В третьем разделе выполняется проектирование архитектуры программного обеспечения, выбор технологий программирования и собственно реализация.

Результаты работы полезны для последующего построения системы.

1 Реновация информационных систем

1.1 Информационные системы

Информационная система (ИС) — это комплекс взаимосвязанных элементов, предназначенных для сбора, хранения, обработки, передачи и использования информации с целью обеспечения эффективного функционирования организации или процесса. [1]

Информационные системы могут быть различного масштаба и сложности. Они могут включать в себя аппаратное обеспечение (компьютеры, серверы, сетевое оборудование), программное обеспечение (операционные системы, базы данных, приложения), данные (информация, хранящаяся и обрабатываемая в системе), процессы (процедуры и алгоритмы работы с данными) и людей (пользователи, администраторы, разработчики).

Информационные системы широко применяются в различных областях, таких как бизнес, наука, образование, здравоохранение, государственное управление и другие. Они помогают автоматизировать бизнес-процессы, обеспечивают доступ к информации, улучшают принятие решений, оптимизируют ресурсы и повышают эффективность работы организаций.

Примеры информационных систем включают в себя:

- Управленческие информационные системы (УИС) - предназначены для поддержки принятия решений на уровне руководства организации. Они обеспечивают анализ данных, формирование отчетов, прогнозирование и планирование;
- Корпоративные информационные системы (КИС) - включают в себя комплексные решения для автоматизации бизнес-процессов внутри организации. Это может быть ERP (Enterprise Resource Planning) система, CRM (Customer Relationship Management) система и другие;
- Информационные системы здравоохранения - применяются в медицинских учреждениях для управления данными пациентов,

расписаниями, заказами лекарств и других аспектов здравоохранения;

- Информационные системы электронной коммерции (ИСЭК) - предоставляют возможность проведения онлайн-торговли, включая обработку заказов, платежи, управление клиентскими данными и т.д;
- Информационные системы управления производством (ИСУП) - используются для контроля и управления производственными процессами, включая планирование производства, управление запасами, отслеживание выполнения заказов и другие функции.

С ростом роли информационных технологий в нашей жизни, информационные системы становятся все более сложными и мощными. Они интегрируют новейшие технологии, такие как искусственный интеллект, большие данные (Big Data), облачные вычисления и интернет вещей (IoT), для обеспечения высокой производительности, безопасности и гибкости.

Однако, помимо преимуществ, информационные системы также представляют ряд вызовов и рисков. К ним относятся вопросы конфиденциальности и безопасности данных, необходимость обучения персонала и обеспечения их надежной работы, а также соответствие правовым и этическим нормам при обработке информации.

В целом, информационные системы являются неотъемлемой частью современного общества. Они облегчают нашу жизнь, улучшают эффективность работы организаций и способствуют развитию новых технологий и инноваций. С развитием технологий информационные системы продолжат развиваться и играть все более важную роль в нашей повседневной жизни.

Однако со временем ИС становятся устаревшими, неэффективными и неспособными удовлетворить новые требования бизнеса и технологические инновации. В таких случаях требуется реновация информационных систем.

1.2 Процесс реновации

Реновация информационных систем — это процесс обновления и улучшения существующих компьютерных систем и программного обеспечения. Это может быть необходимо, если система устарела или не соответствует текущим потребностям компании. Реновация информационных систем может помочь повысить эффективность бизнес-процессов, улучшить безопасность данных и упростить работу сотрудников.

Процесс реновации информационных систем обычно начинается с анализа текущего состояния системы и выявления слабых мест. Это может включать в себя проверку оборудования, программного обеспечения и баз данных, а также оценку системы безопасности и совместимости с другими системами, используемыми в компании.

Определение необходимых изменений и выбор новых технологий и программного обеспечения также является важным этапом реновации информационных систем. Это может включать в себя замену устаревших компонентов, внедрение новых технологий, усовершенствование системы безопасности и обучение сотрудников работе с новыми программами.

Однако при неправильном подходе, процесс реновации может привести к сбоям в работе системы и негативным последствиям для бизнеса. Поэтому для обеспечения надёжной и налаженной работы необходимо составить точную модель системы, на основе которой и будет реализован конечный продукт.

2 Моделирование процесса работы системы

Задачи, с которыми сталкивается человек в своей образовательной, научно-исследовательской и профессиональной деятельности, могут быть разделены на две категории:

– Вычислительные,

– функциональные.

Вычислительные задачи предназначены для выполнения расчетов параметров, характеристик и обработки данных.

Функциональные задачи, в свою очередь, требуют решения при реализации функций управления и проектирования.

Примерами таких задач могут быть управление деятельностью торгового предприятия, организация транспортировки грузов, планирование производственных процессов и тому подобное.

2.1 Моделирование процессов

На рисунке 1 представлена схема, которая иллюстрирует процесс решения задачи с использованием моделирования. В данном контексте, под "реальным объектом" подразумевается объект, который изучается (система, явление, процесс), а "модель" представляет собой физический или виртуальный объект, который замещает реальный объект в процессе познания, одновременно сохраняя его существенные характеристики. "Моделирование" является процессом исследования реального объекта с использованием модели, где исходный объект называется "прототипом" или "оригиналом".

Рисунок 1 – Схема процедуры решения задачи посредством моделирования

Моделирование может применяться не только для воссоздания материальных объектов, но и для описания и изучения различных процессов. Например, инженеры используют аэродинамические трубы для симуляции условий полета самолета на земле. В контексте моделирования, термин "объект моделирования" имеет широкий смысл и может включать как конкретные физические объекты (например, предметы или системы), так и реальные процессы. Модель не воспроизводит все свойства оригинального объекта, а только те, которые необходимы для достижения ее конкретной цели. Поэтому ключевым понятием в моделировании является цель.

Цель моделирования определяет, какие характеристики оригинального объекта должны быть отражены в модели. Другими словами, модель — это упрощенное подобие реального объекта, которое отражает существенные особенности и свойства изучаемого объекта, соответствующие заданной цели моделирования. Моделирование применяется в случаях, когда использование оригинального объекта затруднено или невозможно по разным причинам.

В моделировании акцент делается на выборе свойств объекта, которые необходимы для достижения цели моделирования, а не на воспроизведении всех свойств реального объекта. Модель представляет собой упрощенное

подобие реального объекта, отражающее основные характеристики, необходимые для достижения цели моделирования. Модели используются в случаях, когда использование реального объекта затруднено или невозможно по различным причинам, таким как размеры объекта (слишком большой или маленький), скорость процесса (слишком быстрый или медленный), опасность для окружающей среды, возможность разрушения объекта-оригинала и другие.

Для одного объекта можно создать множество различных моделей, выбор конкретной модели зависит от цели моделирования, определенной в соответствии с поставленной задачей. С другой стороны, одна и та же модель может быть использована для представления разных объектов. Например, математические модели процессов распространения инфекционных болезней и радиоактивного распада могут иметь одинаковое математическое описание.

В моделях существуют общие требования к их свойствам, включая адекватность (точность отображения), конечность (представление оригинала в конечном числе свойств), полноту (предоставление необходимой информации), упрощенность (отображение существенных аспектов объекта), гибкость (способность воспроизведения различных ситуаций) и приемлемая трудоемкость разработки в рамках доступных времени и ресурсов.

2.2 Методы моделирования

Существует множество методов моделирования, которые могут быть применены в различных областях, таких как физика, экономика, биология, компьютерные науки и т.д. Вот некоторые из наиболее распространенных методов моделирования:

- Аналитическое моделирование: В этом методе модель строится на основе математических уравнений и аналитических выражений, которые описывают систему. Этот подход позволяет получить

точные аналитические решения и формализованное описание системы;

- Эмпирическое моделирование: В этом методе модель строится на основе экспериментальных данных, полученных из наблюдений реальной системы. Эмпирические модели могут быть статистическими, регрессионными, нейросетевыми и т.д. Этот подход особенно полезен, когда отсутствует аналитическое понимание системы;
- Агентно - ориентированное моделирование: Этот подход моделирования фокусируется на индивидуальных "агентах" или акторах, которые взаимодействуют друг с другом и окружающей средой. Модель включает правила поведения для каждого агента и может быть использована для изучения эмерджентных свойств и сложных системных эффектов;
- Статистическое моделирование: В статистическом моделировании используются статистические методы для описания и анализа данных. Модели могут включать стохастические компоненты и использоваться для прогнозирования, оценки вероятностей и тестирования гипотез;
- Моделирование на основе машинного обучения: С использованием методов машинного обучения можно построить модель, которая на основе данных обучается предсказывать или классифицировать. Этот подход особенно полезен в задачах распознавания образов, кластеризации данных и других задачах, где требуется обработка больших объемов информации;
- Имитационное моделирование: В этом методе создается компьютерная модель, которая имитирует поведение системы на основе заданных правил и параметров. Симуляции позволяют исследовать системы в различных условиях и проверять гипотезы без фактического проведения реальных экспериментов.

Это лишь несколько примеров методов моделирования, существует множество других подходов и техник, которые могут быть применены в зависимости от конкретной задачи и области применения моделирования.

Для достижения поставленных целей наиболее подходящим является имитационное моделирование, именно он и будет использоваться.

2.3 Имитационное моделирование

Имитационное моделирование (или моделирование событийного потока) — это методология, которая позволяет создавать модели сложных систем и процессов с целью изучения их поведения в контролируемой среде. В имитационном моделировании создается компьютерная модель, которая воспроизводит основные аспекты реальной системы и позволяет исследовать ее динамику, анализировать результаты и принимать решения.

Основные шаги при проведении имитационного моделирования:

- Определение цели моделирования: необходимо определить, что именно нужно изучить или улучшить в системе, также необходимо четко сформулировать цели и ожидаемые результаты;
- Сбор данных: необходимо выполнить сбор информации о системе, которую нужно смоделировать. Это может быть информация о процессах, времени выполнения задач, вероятностных распределениях и других факторах, влияющих на систему;
- Создание модели: на основе собранных данных и целей моделирования нужно разработать компьютерную модель, которая будет имитировать систему. Модель должна содержать все необходимые компоненты и учитывать основные факторы влияния;
- Проверка и корректировка модели: нужно убедиться, что модель работает правильно, путем сравнения результатов моделирования с реальными данными или другими известными моделями. В случае несоответствия необходимо внести корректировки в модель;

– Выполнение экспериментов: необходимо запустить модель на выполнение различных экспериментов, изменяя параметры и условия системы. Также необходимо выполнить сбор данных о поведении системы в различных сценариях и зафиксировать результаты;
– Анализ результатов: необходимо проанализировать полученные данные, сравнить различные сценарии и оценить влияние различных факторов на систему. Также необходимо изучите полученные выводы и сделать соответствующие рекомендации или принимать решения на основе полученной информации.

Имитационное моделирование широко применяется в различных областях, таких как производство, логистика, финансы, здравоохранение и транспорт. Оно позволяет изучать сложные системы, оценивать различные стратегии и оптимизировать процессы без необходимости проведения реальных экспериментов или воздействия на реальные системы. Поэтому и был выбран данный подход.

2.4 Система массового обслуживания

Проведя анализ предметной области, был сделан вывод, что единственной подходящей моделью для поставленной задачи является СМО (Система массового обслуживания).

Системы массового обслуживания (СМО) — это математическая модель, которая используется для анализа и оптимизации процессов обслуживания большого числа клиентов или заявок. Они находят широкое применение в различных областях, таких как транспортные системы, телекоммуникации, производственные процессы, обслуживание клиентов и другие.

СМО состоят из следующих основных компонентов:

- Заявки (клиенты): это объекты, требующие обслуживания. Заявки поступают в систему по времени и могут иметь различные характеристики, такие как время обслуживания, требования к ресурсам и т. д;
- Источники заявок: это источники, генерирующие заявки. Источники могут быть одиночными или множественными, и они могут генерировать заявки в соответствии с определенным распределением, например, пуассоновским процессом;
- Обслуживающие устройства: это ресурсы или системы, предназначенные для обслуживания заявок. Обслуживающие устройства могут быть физическими объектами, такими как серверы, кассы, операторы, или абстрактными единицами времени или обработки;
- Правила обслуживания: это правила, определяющие порядок обслуживания заявок в системе. Некоторые из распространенных правил включают FIFO (первый пришел - первый обслужен), LIFO (последний пришел - первый обслужен) и приоритетное обслуживание.

Целью анализа СМО является определение характеристик процесса обслуживания, таких как среднее время ожидания заявки, вероятность отказа в обслуживании, пропускная способность системы и других показателей. Это позволяет оценить эффективность системы и внести улучшения для оптимизации процесса обслуживания.

Существует множество методов анализа СМО, включая аналитические модели, численное моделирование и имитационное моделирование. Каждый метод имеет свои преимущества и ограничения и выбор метода зависит от конкретной задачи и доступных данных.

Q-схемы являются математическими схемами теории массового обслуживания. Процессы в системах массового обслуживания (СМО) могут иметь различную физическую природу. Это могут быть экономические системы, производственные системы, технические системы, информационные и компьютерные системы.

Характерными особенностями СМО являются:

- случайный характер поступления заявок (требований) на обслуживание;
- случайный характер длительности обслуживания.

Работа СМО включает два необходимых элемента:

- ожидание обслуживания;
- собственно обслуживание, которое предполагает наличие прибора обслуживания.

Различные типы СМО отличаются друг от друга по следующим параметрам:

1. по типу ожидания:

- время ожидания не ограничено;
- время ожидания ограничено;
- СМО без ожидания (если канал закрыт, то заявка не принимается).

2. по типу постановки в очередь:
– по очереди в порядке поступления;
– с учётом приоритета.
3. по типу каналов обслуживания:
– одноканальные СМО;
– многоканальные СМО.
4. по длительности обслуживания:
– с неограниченным временем обслуживания;
– с ограниченным временем обслуживания.

Существует множество типов СМО, вот некоторые из них:

– Система обслуживания M/M/1: это одноканальная система обслуживания, в которой заявки поступают согласно пуассоновскому процессу (M - экспоненциальное распределение интервалов между поступлением заявок) и обслуживаются одним обслуживающим устройством. Время обслуживания также имеет экспоненциальное распределение. Эта модель широко используется для анализа простых систем обслуживания;
– Система обслуживания M/M/c: это система обслуживания с несколькими обслуживающими устройствами (c) в параллель. Заявки также поступают согласно пуассоновскому процессу, а время обслуживания имеет экспоненциальное распределение. Эта модель используется для анализа систем с несколькими параллельными ресурсами, такими как кассы в супермаркете или серверы в компьютерной сети;
– Система обслуживания M/G/1: В этой модели время обслуживания имеет произвольное распределение (G), а заявки все еще поступают согласно пуассоновскому процессу. Такие системы встречаются,

например, в случае, когда время обслуживания зависит от сложности заявки или от производительности обслуживающего устройства.

– Система обслуживания G/G/c: это модель, в которой и время обслуживания, и интервалы между поступлением заявок имеют произвольные распределения. Системы такого типа сложнее анализировать, и для их моделирования могут использоваться численные или имитационные методы;
– Система обслуживания с приоритетами: В этом типе системы заявки обслуживаются с учетом приоритетов, установленных для каждой заявки.

Рассмотрим моделирование входного потока. Пусть начальный момент времени $t_0 = 0$, и в случайные моменты $\{t_1, t_2, \dots, t_n, \dots\}$ поступают заявки. Заметим, что промежуток между поступлениями заявок $\Delta t_i = t_{i+1} - t_i$, естественно, тоже случайная величина.

Поток называется однородным, если поступают заявки одного типа. Такой поток полностью характеризуется набором $\{t_o, t_1, t_2, \dots, t_n, \dots\}$, если поступают заявки с различным набором признаков, то такой поток называется неоднородным.

На рисунке 2 приведена схема работы одноканальной СМО.

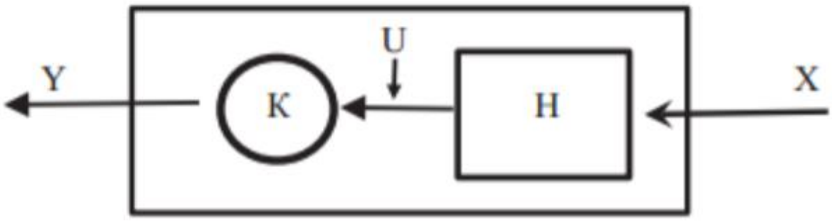

Рисунок 2 – Схема работы одноканальной СМО

Здесь X – входной поток, H – накопитель с заданными параметрами (например, с заданной емкостью C_n), U - поток обслуживания,

К – канал обслуживания, Y – выходной поток. Таким образом, модель СМО есть совокупность $Q = (X, H, U, Z, A)$, где $Z = \{z^H, z^K\}, z^H$ – число заявок в накопителе, z^K – число заявок в канале обслуживания. Обычно $z^K = 0$, если в канале нет заявок, и $z^K = 1$, если заявки есть. А - алгоритмы обслуживания.

Из проведённого анализа предметной области было установлено, что по своему характеру и поведению систему можно квалифицировать как одноканальную СМО с накопителем ограниченной ёмкости, но неограниченной очередью, с неограниченным ожиданием и без отказов.

2.5 Математическая модель

В качестве математической модели используется одноканальная СМО с накопителем ограниченной ёмкости, но неограниченной очередью, с неограниченным ожиданием и без отказов. В одноканальных системах массового обслуживания с неограниченной очередью длина очереди (теоретически) может расти до бесконечности. Примером такой системы массового обслуживания может служить очередь в хлебном магазине (булочной) с одним кассиром. Граф одноканальной системы без ограничений на длину очереди изображён на рис. 3.

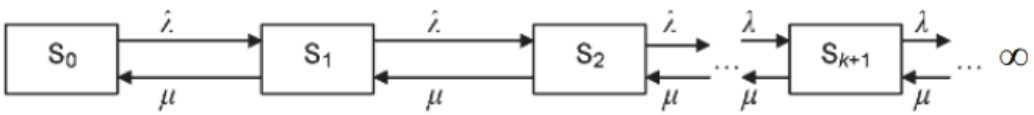

Рисунок 3 - Граф состояний одноканальной системы массового обслуживания с неограниченной очередью

Возможные состояния системы будут следующие:

- s_0 - канал обслуживания свободен;
- s_1 - канал обслуживания занят, но очереди нет;
- s_2 - канал обслуживания занят, в очереди одна заявка;
- s_{k+1} - канал обслуживания занят, в очереди k заявок.

Поскольку в системах массового обслуживания отсутствует ограничение на длину очереди, то любая заявка может быть обслужена, т.е. относительная пропускная способность равна

$$Q = P_{\text{обсл}} = 1; \quad (1)$$

Вероятность отказа в такой системе невозможна, поэтому принимается как $P_{\text{отк}} = 0$.

Абсолютная пропускная способность

$$A = \lambda * Q = \lambda; \quad (2)$$

Вероятность пребывания в очереди (k) заявок

$$P_k = p^k(1-p); \quad (3)$$

Среднее число заявок в очереди

$$L_{\text{оч}} = \frac{p^2}{1-p}; \quad (4)$$

Среднее число заявок в системе

$$\mathrm{L}_{\text{смо}} = L_{\text{оч}} + p = \frac{p^2}{1-p} + p = \frac{p}{1-p}; \quad (5)$$

Среднее время ожидания обслуживания в очереди

$$T_{\text{оч}} = \frac{p}{\mu(1-p)} = \frac{p^2}{1-p} = \frac{L_{\text{оч}}}{\lambda}; \quad (6)$$

Среднее время пребывания заявки в системе

$$T_{\text{смо}} == T_{\text{оч}} + t_{\text{обсл}} = \frac{L_{\text{смо}}}{\lambda}; \quad (7)$$

3 Проектирование и реализация имитационной модели

3.1 Обзор и выбор технологий программирования

Технологии программирования — это методы и инструменты, которые используются для создания программного обеспечения. Существует множество различных технологий, и каждая из них имеет свои особенности и применение.

Некоторые из самых популярных технологий программирования включают в себя:

- Языки программирования. Языки программирования определяют синтаксис и семантику, которые используются для написания кода. Существует множество языков программирования, включая Java, Python, C++, JavaScript и другие;
- Интегрированные среды разработки (IDE). IDE — это приложения, которые облегчают процесс разработки программного обеспечения, предоставляя программисту средства для написания, отладки и тестирования кода в одном месте. Примеры IDE включают в себя Visual Studio, Eclipse и IntelliJ IDEA.

3.1.1 Обзор и выбор языка программирования

Существует множество языков программирования, каждый из которых предназначен для решения определенных задач и обладает своими особенностями. Вот некоторые из наиболее популярных языков программирования:

- Python: Простой и понятный язык программирования, широко используется для различных целей, включая веб-разработку, научные вычисления, анализ данных и автоматизацию задач;

- JavaScript: Язык программирования, который применяется в основном для разработки интерактивных веб-страниц и приложений. Он работает в браузере и может использоваться и на серверной стороне с использованием Node.js;
- Java: Универсальный язык программирования, используемый для создания различных типов приложений, включая мобильные приложения, веб-приложения и корпоративные системы;
- C#: Язык программирования, разработанный Microsoft, используется для создания приложений для платформы .NET, включая Windows-приложения, веб-приложения и игры;
- Ruby: Простой и элегантный язык программирования, который активно используется в веб-разработке с использованием фреймворка Ruby on Rails;
- PHP: Язык программирования, специализирующийся на разработке веб-приложений и динамических веб-сайтов;
- Swift: Язык программирования, разработанный Apple для создания приложений для iOS, macOS, watchOS и tvOS;
- Go: Компилируемый язык программирования, созданный Google, который обладает высокой производительностью и предназначен для создания распределенных систем;
- Rust: Безопасный и быстрый язык программирования, призванный решить проблемы с памятью и безопасностью, часто используется для системного программирования и разработки низкоуровневых приложений;
- C++: Мощный язык программирования, который широко применяется для разработки системного программного обеспечения, игр, мобильных приложений и производительных приложений.

Это только небольшой список языков программирования, существуют и другие языки, такие как Kotlin, TypeScript, Perl, и т. д. Выбор языка программирования зависит от задачи, которую планируется решить, и от предпочтений и опыта.

Исходя из поставленной задачи, а также имеющихся знаний и навыков, для реализации поставленной задачи был выбран язык программирования C++.

3.1.2 Обзор и выбор среды разработки

Среда разработки (Integrated Development Environment, IDE) представляет собой программное обеспечение, которое облегчает процесс создания, отладки и развертывания программного кода. Вот некоторые из популярных сред разработки:

- Visual Studio Code: Разработана компанией Microsoft, является одной из самых мощных и полнофункциональных IDE для различных языков программирования, включая C++, C#, Visual Basic, Python, JavaScript и другие;
- Eclipse: Бесплатная и открытая среда разработки, предназначенная для различных языков программирования, таких как Java, C/C++, Python и других. Она также поддерживает расширения для разработки веб-приложений и других технологий;
- IntelliJ IDEA: предоставляет поддержку разработки на различных языках программирования, включая Java, Kotlin, Scala, JavaScript и другие. Он предлагает множество продвинутых функций и инструментов для повышения производительности разработчиков;
- PyCharm: Среда разработки, специализирующаяся на работе с языком программирования Python. Предлагает возможности для разработки веб-приложений, научных вычислений и других задач, связанных с Python;

- Xcode: IDE, разработанная Apple для создания приложений для платформ iOS, macOS, watchOS и tvOS. Она включает в себя множество инструментов и ресурсов для разработки мобильных и десктопных приложений для устройств Apple;
- Android Studio: Официальная IDE для разработки приложений под платформу Android. Предлагает множество инструментов для создания пользовательского интерфейса, отладки, сборки и тестирования приложений для Android;
- Visual Studio Code: Легковесная и мощная среда разработки, поддерживающая различные языки программирования. Она обладает множеством расширений и настраиваемых возможностей, что делает ее популярной среди разработчиков;
- NetBeans: Платформонезависимая среда разработки, поддерживающая различные языки программирования, такие как Java, PHP, C/C++ и другие. Предлагает набор инструментов для разработки приложений различного типа;
- Sublime Text: Быстрый и легкий редактор кода, поддерживающий множество языков программирования. Обладает мощными функциями редактирования и настраиваемыми возможностями;
- Atom: Бесплатный и расширяемый редактор кода, разработанный GitHub. Предлагает интуитивный интерфейс, поддержку различных языков программирования и настраиваемые возможности.

Из-за имеющегося опыта работы в среде разработки Visual Studio Code был выбран именно он, ниже произведён более подробный обзор данной IDE.

Visual Studio Code (VS Code) — это бесплатная среда разработки с открытым исходным кодом, разработанная Microsoft. Она предоставляет мощные инструменты для написания и отладки кода, а также поддерживает множество языков программирования и платформ.

Особенности Visual Studio Code:

- Расширяемость: VS Code имеет обширную экосистему расширений, которые позволяют настраивать среду разработки под ваши потребности. Вы можете установить расширения для поддержки конкретных языков, инструментов сборки, отладчиков и многого другого;
- Интегрированный отладчик: Встроенный отладчик в VS Code позволяет вам отслеживать выполнение вашего кода, устанавливать точки останова, проверять значения переменных и многое другое;
- Автодополнение и подсветка синтаксиса: Среда поддерживает автодополнение кода, что помогает ускорить процесс написания кода и предотвращает ошибки. Также в VS Code доступна подсветка синтаксиса для множества языков программирования;
- Встроенная система контроля версий: VS Code имеет интеграцию с различными системами контроля версий, такими как Git. Вы можете просматривать изменения, сравнивать файлы и управлять репозиторием прямо из среды разработки;
- Разделение на панели: Среда позволяет разделять окно на несколько панелей, что полезно при работе с несколькими файлами или просмотре разных частей вашего проекта одновременно;
- Поддержка распространенных языков программирования: Visual Studio Code поддерживает множество языков программирования, включая JavaScript, Python, C++, Java, HTML, CSS и многие другие;
- VS Code доступен на различных платформах, включая Windows, macOS и Linux. Он обладает активным сообществом пользователей и разработчиков, что обеспечивает постоянную поддержку и обновления.

3.2 Проектирование архитектуры программного обеспечения

На основе проведённого анализа предметной области, и построенной математической модели, была спроектирована архитектура программного обеспечения. Блок-схема полученной архитектуры представлена на рисунке 4.

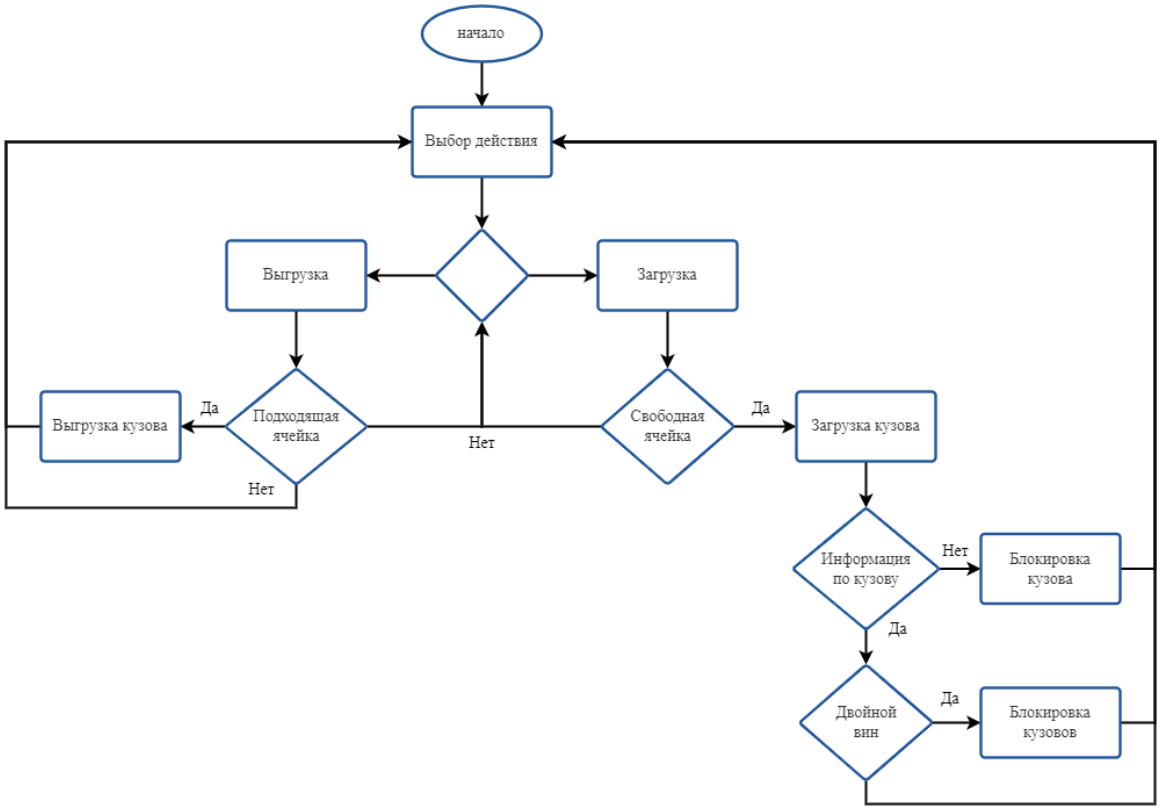

Рисунок 4 – Блок-схема алгоритма

3.3 Реализация программы

Для работы со стандартными функциями ввода и вывода подключается библиотека iostream, а также определяется пространство имён std.

На рисунке 5 показано подключение библиотеки iostream и определение пространства имён std.

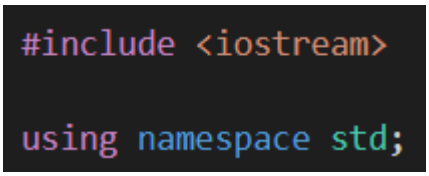

Рисунок 5 – Подключение библиотеки и определение пространства имён

На рисунке 6 изображено объявление и инициализация переменных, которые содержат информацию об объёме данных.

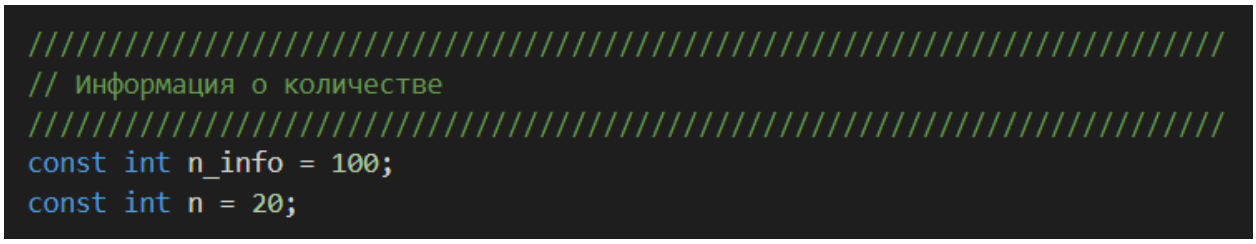

Рисунок 6 – Определение тестовой информации

В накопителе ячейка может иметь 3 состояния:

– Со значением 0 – свободная и не заблокированная. В неё может быть загружен кузов;
– Со значением 1 – присутствует кузов и не заблокирована. Из неё кузов может быть выгружен;

– Со значением 3 – ячейка заблокирована вне зависимости от того есть в не кузов или нет, тем самым ограничивается загрузка и выгрузка из неё.

Их объявление представлено на рисунке 7.

```
////////////////////////////////////////////////////////////////////////////
// Типы состояний ячейки
////////////////////////////////////////////////////////////////////////////
#define empty 0   // Свободная
#define unblock 1 // Есть кузов и не заблокирована
#define block 2   // Заблокирована
```

Рисунок 7 – Состояния ячеек

Для работы хранения данных используются структуры информации. Сначала они просто объявляются, а позже создаётся массив структур на нужный объём информации.

Структура Information будет содержать информацию о кузовах с других систем. Информация включает в себя vin, чертёж и номер. Данная структура представлена на рисунке 8.

```
////////////////////////////////////////////////////////////////////////////
// Структура информации о кузовах
////////////////////////////////////////////////////////////////////////////
struct Information
{
    int vin;
    int cherteg;
    int number;
};
```

Рисунок 8 – Структура информации о кузовах

Структура Sklad будет содержать в себе информацию о ячейках на складе, а то пустая я ли ячейка или нет, какое состояние имеет. Информация включает в себя vin, чертёж, номер, а также состояние ячейки. Структура Sklad показана на рисунке 9.

```
////////////////////////////////////////////////////////////////////////
// Структура информация о ячейках склада
////////////////////////////////////////////////////////////////////////
struct Sklad
{
    int vin;
    int cherteg;
    int number;
    int state;
};
```

Рисунок 9 – Структура информации о ячейках склада

Структура cell будет содержать информацию о задании на выгрузку или загрузку. Например, при загрузке в неё записывается следующая информация: номер ячейки куда можно загрузить кузов и номер ячейки информации, из которой она будет загружена, а при выгрузке в неё записывается номер ячейки, из которой следует выгрузить кузов. Структура Cell изображена на рисунке 10.

```
////////////////////////////////////////////////////////////////////////
// Структура промежуточной информации
////////////////////////////////////////////////////////////////////////
struct Cell
{
    int info_number;
    int load_number;
    int unload_number;
};
```

Рисунок 10 – Структура промежуточной информации

При создании задания на загрузку в первую очередь происходит поиск свободной и не заблокированной ячейки, в которую можно загрузить кузов.

Поиск происходит по всему складу, перебирая ячейки и их состояния, если такая ячейка найдена, то в структуру cell записывается номер этой ячейки и функция возвращает 1, как признак того, что подходящая ячейка найдена, иначе выводится информация, что такие ячейки отсутствуют и возвращается признак 0. За данные действия отвечает функция GetEmptyCell(), которая изображена на рисунке 11.

```
////////////////////////////////////////////////////////////////////////
// Поиск свободной ячейки
////////////////////////////////////////////////////////////////////////
bool GetEmptyCell(Sklad *sklad, Cell *cell)
{
    cout << "Поиск свободной ячейки -> ";
    for(int i = 0; i < n; i++)
    {
        if(sklad[i].state == empty)
        {
            cout << "Свободная ячейка найдена ->";
            cell->load_number = i;
            return 1;
        }
    }
    cout << "Свободных ячеек нет!";
    return 0;
}
```

Рисунок 11 – Функция поиска свободной ячейки

Void LoadVin() – это функция, которая отвечает за загрузку кузова в ячейку, приставка void означает что, функция никакую информацию не возвращает, только выполняет определённые действия. Объявляется переменная n и ей присваивается информация о номере ячейки, в которую

следует загрузить кузов, кузов загружается и ячейке присваивается статус 1, то есть в ячейке присутствует кузов, и он доступен на выгрузку. (рисунок 12)

```
////////////////////////////////////////////////////////////////////////
// Загрузка кузова в ячейку
////////////////////////////////////////////////////////////////////////
void LoadVin(Sklad *sklad, Cell *cell)
{
    int n = cell->load_number;
    sklad[n].state = unblock;
    cout << " Кузов загружен в ячейку № " << n << endl;
}
```

Рисунок 12 – Функция загрузки кузова в ячейку

Функция bool SearchVinInformation() – это функция, которая отвечает за поиск информации о кузове в других системах. Если информация найдена, то выводится сообщение, что информация найдена, в структуру cell записывается адрес ячейки информации, которой она находится, чтобы её оттуда загрузить, и возвращается признак 1, как ответ об успешном поиске. Если информация не найдена, то выводится соответствующая информация и далее уже система предпринимает нужные действия. Функция bool SearchVinInformation() представлена на рисунке 13.

```
////////////////////////////////////////////////////////////////////////
// Поиск информации по кузову
////////////////////////////////////////////////////////////////////////
bool SearchVinInformation(Information *info, Cell *cell, int vin)
{
    cout << "Поиск информации по вин -> ";
    for(int i = 0; i < n_info; i++)
    {
        if(info[i].vin == vin)
        {
            cout << "Информация по вин найдена -> ";
            cell->info_number = i;
            return 1;
        }
    }
    cout << "Информация по вин не найдена -> ";
    return 0;
}
```

Рисунок 13 – Функция поиска информации по кузову

Функция void LoadVinInformation() – выполняет загрузку найденной информации о кузове и дополнительно присваивает ячейки, в которую загружен кузов состояние 1, то есть содержащую кузов и доступную на выгрузку (рисунок 14).

```
////////////////////////////////////////////////////////////////////////
// Загрузка информации по кузову
////////////////////////////////////////////////////////////////////////
void LoadVinInformation(Sklad *sklad, Information *info, Cell *cell)
{
    int out = cell->info_number;
    int in = cell->load_number;
    cout << "Загрузка информации по вин " <<  << endl;
    sklad[in].vin = info[out].vin;
    sklad[in].cherteg = info[out].cherteg;
    sklad[in].number = info[out].number;
    sklad[in].state = unblock;
}
```

Рисунок 14 – Функция загрузки информации по кузову

Помимо того, что кузов может быть заблокирован из-за того, что он не известен, ещё и из-за того, что на складе уже имеется кузов с таким же vin, что говорит о том, что где-то есть несоответствие и до разбирательств эти кузова должны быть заблокированы. Для этого есть функции SearchDoubleVin и BlockDoubleVin.

Функция bool SearchDoubleVin() – занимается поиском одинаковых винов на складе, если такой вин найден, то выводится сообщение об этом и возвращается 1, как признак того, что такие кузова найдены. Иначе выводится сообщение, что такие кузова отсутствуют. Данная функция представлена на рисунке 15.

```
////////////////////////////////////////////////////////////////////////////
// Поиск двойного вина
////////////////////////////////////////////////////////////////////////////
bool SearchDoubleVin(Sklad *sklad, int vin)
{
    int total = 0;
    cout << "Поиск двойного вин -> ";
    for(int i = 0; i < 2; i++)
    {
        if(vin == sklad[i].vin && total > 1)
        {
            total++;
        }
        if(total > 1)
        {
            cout << "Найден двойной вин" << endl;
            return 1;
        }
    }
    cout << "Двойной вин отсутствует";
    return 0;
}
```

Рисунок 15 – Функция поиска двойного вина

Функция BlockDoubleVin() – выполняется блокировку списка кузовов на складе с одинаковым вин, блокировка происходит изменением их статуса на 2, до дальнейших выяснений (рисунок 16).

```
////////////////////////////////////////////////////////////////////////
// Блокировка одинаковых винов
////////////////////////////////////////////////////////////////////////
void BlockDoubleVin(Sklad *sklad, int vin)
{
    for(int i = 0; i < n; i++)
    {
        if(sklad[i].vin == vin)
        {
            sklad[i].state = block;
        }
    }
}
```

Рисунок 16 – Функция блокировки одинаковых винов

Функция void UnknowVin() – выполняет блокировку кузова с неизвестным вин, до выяснения подробностей. Сначала загружает информацию о номере ячейки, в которой этот кузов находится, а далее меняет её статус на 2 и выводит сообщение о том, что ячейка заблокирована. Эта функция представлена на рисунке 17.

```
////////////////////////////////////////////////////////////////////////
// Блокировка неизвестного вина
////////////////////////////////////////////////////////////////////////
void UnknownVin(Sklad *sklad, Cell *cell)
{
    int n = cell->load_number;
    sklad[n].state = block;
    cout << "Ячейка № " << n << " заблокирована" << endl;
}
```

Рисунок 17 – Блокировка неизвестного вина

Основная модель процесса работы системы описана в функция zahod и vihod.

За задание загрузки кузова на склад отвечает функция Zahod. Её работа происходит следующим образом: выполняется поиск свободной и незаблокированной ячейки, если таковая найдена то происходит загрузка в неё, далее совершается поиск информации об этом кузове в других системах, если информация найдена, то происходит её загрузка и далее выполняется поиск двойных вин, если они найдены, то происходит их блокировка. Если информация о кузове не была найдена, то выполняется блокировка ячейки, в которую она была загружена. Если изначально доступная ячейка не была найдена, то работа функции прекращается и ожидается новое задание. Программное построение модели захода кузова представлена на рисунке 18.

```
////////////////////////////////////////////////////////////////////////////
// Заход кузова
////////////////////////////////////////////////////////////////////////////
void Zahod(Sklad *sklad, Information *info, Cell *cell, int new_vin)
{
    // Поиск свободной ячейки
    if(GetEmptyCell(sklad, cell) == 1)
    {

        // Загрузка кузова на склад
        LoadVin(sklad, cell);

        // Поиск информации о вин
        if(SearchVinInformation(info, cell, new_vin) == 1)
        {
            // Загрузка информации о вин
            LoadVinInformation(new_vin, sklad, info);

            // Поиск двойных вин
            if(SearchDoubleVin(sklad, new_vin) == 1)
            {
                // Блокировка двойных вин
                BlockDoubleVin(sklad, new_vin);
            }
        }
        else
        {
            // Действие с неизвестным вин
            UnknownVin(sklad, cell);
        }
    }
    else
    {
        return;
    }
}
```

Рисунок 18 – Функция, имитирующая заход кузова

При задании выгрузки кузова, происходит поиск наиболее подходящего, а то есть доступный на выгрузку и имеющий минимальный номер.

bool SearchUnloadVin() - функция, которая выполняет поиск кузова на выгрузку. Просходит это следующим образом, выполняется перебор ячеек на складе, по критериям, что там имеется кузов и он не заблокирован, а так же сравнивается его номер с предыдущим найденным, какой из них меньше он и становится претендентом на выгрузку. Конкретный подходящий кузов определяется только после перебора всех ячеек склада. Если такой кузов найден, то выводится сообщение, что найден такой вин и номер ячейки, в которой он находится записывается в структуру cell, а возвращается 1, как признак успешного поиск. Если такой кузов не найден, то выводится соответствующее сообщение и возвращается признак 0. Реализация этой функции показана на рисунке 19.

```
////////////////////////////////////////////////////////////////////////////
// Поиск кузова на выгрузку
////////////////////////////////////////////////////////////////////////////
bool SearchUnloadVin(Sklad *sklad, Cell *cell)
{
    int min = n;
    cout << "Поиск вин на выгрузку -> ";

    for(int i = 0; i < n; i++)
    {
        if(sklad[i].state != empty && sklad[i].state != block)
        {
            if(sklad[i].number < min)
            {
                min = sklad[i].number;
            }
        }
    }

    if(min < n)
    {
        cout << "Найден вин на выгрузку" << endl;
        cell->unload_number = min;
        return 1;
    }

    else
    {
        cout << "Такой вин отсутствует" << endl;
        return 0;
    }
}
```

Рисунок 19 – Функция поиска кузова на выгрузку

Функция void UnloadVin() – выполняет выгрузку кузова из ячейки (рисунок 20).

```
////////////////////////////////////////////////////////////////////////
// Выгрузка кузова из ячейки
////////////////////////////////////////////////////////////////////////
void UnloadVin(Sklad *sklad, Cell *cell)
{
    int n = cell->unload_number;
    cout << " Кузов выгружен из ячейки № " << n << endl;
}
```

Рисунок 20 – Функция выгрузки кузова из ячейки

За задание выгрузки кузова со склада отвечает функция Vihod. Её работа происходит следующим образом: происходит поиск подходящей ячейки, если таковая найдена, то выполняется её выгрузка иначе работа функции прекращается и программа переходит в режим ожидания задания. Программное построение этой модели представлено на рисунке 21.

```
////////////////////////////////////////////////////////////////////////
// Выход кузова
////////////////////////////////////////////////////////////////////////
void Vihod(Sklad *sklad, Cell *cell)
{
    // Поиск походящего кузова на выгрузку
    if(SearchUnloadVin(sklad, cell) == 1)
    {
        // Выгрузка
        UnloadVin(sklad, cell);
    }
    else
    {
        return;
    }
}
```

Рисунок 21 – Функция, имитирующая выход кузова

Как говорилось ранее, в структуре cell хранится информации и номерах ячеек на загрузку и выгрузку, а также об адресе ячейки информации, в которой находится информация о кузове. Чтобы данные не смешивались и не было сбоев, следует очищать структуру cell. Для это реализована функция void Ochistka_info(), изображённая на рисунке 22.

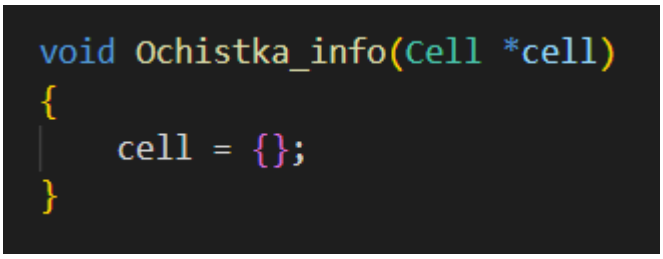

Рисунок 22 – Функция очистки структуры Cell

Заключение

В рамках выпускной квалификационной работы была разработана модель процесса работы системы “Накопитель окрашенных кузовов”.

В ходе выполнения выпускной квалификационной работы были решены следующие задачи:

- проанализирована предметная область;
- рассмотрены методы моделирования и выбран наиболее подходящий;
- составлена математическая модель системы;
- рассмотрены технологии программирования и выбраны наиболее подходящие;
- спроектирована архитектура программного обеспечения;
- реализовано программное обеспечение;
- протестирована разработанная программа.

При разработке модели процесса работы системы накопителя окрашенных кузовов были учтены следующие аспекты:

- Объем и вместимость: Модель учитывает вместимость накопителя, то есть сколько окрашенных кузовов он может содержать одновременно. Это важно для оптимизации процесса и планирования загрузки и выгрузки кузовов;
- Управление и мониторинг: Модель включает систему управления и мониторинга накопителя. Она позволяет контролировать заполнение и освобождение мест для кузовов, отслеживать их текущее положение в системе, а также предоставляет информацию о доступных кузовах и статусе каждого из них;
- Безопасность: Модель учитывает вопросы безопасности при работе с накопителем окрашенных кузовов. Это может включать меры предосторожности при перемещении кузовов внутри системы,

предотвращение повреждений покрытия и обеспечение безопасности персонала, работающего с накопителем;

- Оптимизация процесса: Модель помогает оптимизировать процесс работы накопителя окрашенных кузовов. Она может предлагать оптимальные последовательности загрузки и выгрузки кузовов, сокращать время простоя и максимизировать использование доступного пространства в накопителе;
- Интеграция с другими системами: Модель учитывает возможность интеграции с другими системами, такими как система окраски кузовов или система управления производством. Это позволяет обеспечить согласованность и эффективность работы всего производственного процесса.

Результатом работы является созданная программа на основе разработанной модели, описывающая процесс работы реального производственного цеха. Данная модель была использована для построения новой системы. Новая система уже разработана, прошла необходимые проверки и была внедрена в производство. С момента внедрения система безотказно работает и выполняет свои функции.

Список используемой литературы и используемых источников

1. Информационные технологии и информационные системы [Электронный источник]. Режим доступа https://intuit.ru/studies/courses/3735/977/lecture/14671, дата доступа 05.06.2023.
2. Теория массового обслуживания [Электронный источник]. Режим доступа https://elar.urfu.ru/bitstream/10995/117140/1/978-5-7996-3539-8_2022.pdf, дата доступа 06.06.2023.
3. Алгазинов Э. К., Сирота А. А. Анализ и компьютерное моделирование информационных процессов и систем; Диалог-МИФИ - , 2009. - 416 с.
4. Алешин Л. И. Обеспечение автоматизированных библиотечных информационных систем (АБИС); Форум - Москва, 2012. - 432 с.
5. Антопольский Александр Борисович Интеграция Библиотечных И Архивных Информационных Систем; New York: HarperBusiness - Москва, 2012. - 562 с.
6. Васильев Р. Б., Калянов Г. Н., Левочкина Г. А. Управление развитием информационных систем; Горячая Линия - Телеком - , 2009. - 378 с.
7. Воройский Ф. С. Основы проектирования автоматизированных библиотечно-информационных систем; ФИЗМАТЛИТ - Москва, 2002. - 384 с.
8. Грекул В. И., Денищенко Г. Н., Коровкина Н. Л. Управление внедрением информационных систем; Интернет-университет информационных технологий, Бином. Лаборатория знаний - Москва, 2008. - 224 с.
9. Емельянова Н. З., Партыка Т. Л., Попов И. И. Проектирование информационных систем; Форум - Москва, 2009. - 432 с.
10. Емельянова Н. З., Партыка Т. Л., Попов И. И. Устройство и функционирование информационных систем; Форум, Инфра-М - Москва, 2012. - 448 с.

11. Заботина Н. Н. Проектирование информационных систем; Инфра-М - , 2013. - 336 с.

12. Акопов, А.С. Имитационное моделирование: Учебник и практикум для академического бакалавриата / А.С. Акопов. - Люберцы: Юрайт, 2016. - 389 с.

13. Булыгина, О.В. Имитационное моделирование в экономике и управлении: Учебник / О.В. Булыгина, А.А. Емельянов, Н.З. Емельянова. - М.: Инфра-М, 2017. - 447 с.

14. Варжапетян, А.Г. Имитационное моделирование на GPSS/H / А.Г. Варжапетян. - М.: Вузовская книга, 2007. - 424 с.

15. Вьюненко, Л.Ф. Имитационное моделирование: Учебник и практикум для академического бакалавриата / Л.Ф. Вьюненко, М.В. Михайлов, Т.Н. Первозванская. - Люберцы: Юрайт, 2016. - 283 с.

16. Девятков, В.В. Имитационное моделирование: Учебное пособие / Н.Б. Кобелев, В.А. Половников, В.В. Девятков. - М.: КУРС, НИЦ Инфра-М, 2013. - 368 с.

17. Карпов, Ю. Имитационное моделирование систем. Введение в моделирование с AnyLogic 5 / Ю. Карпов. - СПб.: BHV, 2009. - 400 с.

18. Кобелев, Н.Б. Имитационное моделирование: Учебное пособие / Н.Б. Кобелев, В.В. Девятков, В.А. Половников. - М.: Инфра-М, 2016. - 448 с.

19. Директор, С. Введение в теорию систем / С. Директор, Р. Рорер. - М.: Мир, 2016. - 464 с.

20. Исследования по общей теории систем. - М.: Прогресс, 2015. - 520 с.

21. Карташевский, В. Г. Основы теории массового обслуживания. Учебник / В.Г. Карташевский. - М.: Горячая линия - Телеком, 2018. - 132 с.

22. Карташевский, В.Г. Основы теории массового обслуживания / В.Г. Карташевский. - М.: Радио и связь, 2016. - 108 с.

23. Карташевский, Вячеслав Григорьевич Основы теории массового обслуживания. Учебник для вузов / Карташевский Вячеслав Григорьевич. - М.: Горячая линия - Телеком, 2017. - 449 с.

24. Кениг, Д. Методы теории массового обслуживания / Д. Кениг, Д. Штойян. - Москва: ИЛ, 2016. - 128 с.

25. Кирпичников, А.П. Методы прикладной теории массового обслуживания / А.П. Кирпичников. - Москва: РГГУ, 2018. - 336 с.

26. Кофман, А. Массовое обслуживание. Теория и приложения / А. Кофман, Р. Крюон. - М.: Мир, 2016. - 302 с.

27. Коэн, Дж. Граничные задачи в теории массового обслуживания / Дж. Коэн, О. Боксма. - М.: Мир, 2019. - 272 с.