МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Институт перспективных технологий и индустриального программирования (ИПТИП)

Кафедра индустриального программирования

КУРСОВОЙ ПРОЕКТ (РАБОТА)

по дисциплине

Технологии индустриального программирования

Тема курсового проекта (работы) Разработка программного продукта моделирования информационно-управляющей системы на языке C++

Студент группы Калинин Н.В., ЭФБО-09-23 _____
(Ф.И.О., учебная группа) *(подпись студента)*

Руководитель курсового проекта (работы) Соловьев А.М., доцент, к.т.н. _____
(Ф.И.О., должность, звание, ученая степень) *(подпись руководителя)*

Курсовой проект (работа) представлен(а) к защите «31» мая 2024 г.
Допущен(а) к защите «31» мая 2024 г.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА - Российский технологический университет»
РТУ МИРЭА

Институт перспективных технологий и индустриального программирования (ИПТИП)

Кафедра индустриального программирования

Утверждаю
Заведующий кафедрой индустриального программирования
_____ / Юдин А.В.
(подпись) *(Ф.И.О.)*
29 февраля 2024 г.

ЗАДАНИЕ
на выполнение курсового проекта (работы)
по дисциплине
Технологии индустриального программирования

Тема курсового проекта (работы) Разработка программного продукта моделирования информационно-управляющей системы на языке C++

Студент Калинин Н.В. Группа ЭФБО-09-23

Исходные данные: Индивидуальный вариант на курсовую работу №278

Перечень вопросов подлежащих разработке и обязательного материала:
Провести теоретический обзор области разработки программного продукта.
Разработать технологический стек, соответствующий предметной области.
Разработать программный продукт по теме и протестировать его.
4. Подготовить отчет по курсовой работе и представить его к защите.

Срок представления к защите курсового проекта (работы) до 31 мая 2024 г.

Задание на курсовой проект (работу) выдал _____ Соловьев А.М.
подпись руководителя) Ф.И.О. руководителя)
29 февраля 2024 г.

Задание на курсовой проект (работу) получил _____ Калинин Н.В.
подпись обучающегося) Ф.И.О. обучающегося)
29 февраля 2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современный рынок разработки программного обеспечения обязывает специалистов не только быть экспертами в своей области, но и ориентироваться в широком спектре смежных технологических областей. Этот проект направлен на создание программного продукта для моделирования информационно-управляющей системы, автоматизирующей процессы на участке конвейерного производства печатных плат. Использование языка программирования C++ обеспечивает ряд преимуществ, включая высокую производительность, эффективное управление ресурсами системы, широкие возможности по объектно-ориентированному программированию, а также обширную базу библиотек и инструментов разработки. Такая деятельность, помимо накопления практического опыта, способствует формированию нового взгляда на классические задачи разработки программных комплексов и программного обеспечения.

Цель работы: разработка программного продукта для моделирования информационно-управляющей системы автоматизации участка конвейерного производства печатных плат на языке программирования C++.

Задачи работы

1. Описать программируемую систему.
2. Рассмотреть существующие решения-аналоги по данной, либо смежной темам.
3. Сформировать требования к программируемой системе.
4. Спроектировать диаграмму состояний системы.
5. Спроектировать диаграмму классов системы.
6. Спроектировать диаграмму последовательности для системы.
7. Реализовать программный продукт в соответствии с требованиями.
8. Провести функциональное тестирование программного продукта.
9. Составить инструкцию по использованию программного продукта.
10. Составить отчет по работе.

11. Сдать отчет и представить его к защите.

Объектом данного исследования является проектирование и разработка информационно-управляющей системы для оптимизации производственных процессов на предприятии с использованием языка программирования C++.

Предметов исследования в данной работе является проектирование и разработка программного продукта на языке C++, который будет моделировать информационно-управляющую систему для эффективной автоматизации производственного участка.

В качестве основных методов исследования применены анализ, синтез, сравнение и моделирование. Практическая реализация поставленной задачи соответствует основным подходам к разработке программного обеспечения.

Информационной базой исследования являются открытые источники, в том числе доступные в сети Интернет, а также материалы курса «Технологии индустриального программирования», доступные через систему дистанционного обучения РТУ МИРЭА.

В данном отчете будет представлен процесс разработки программного продукта, в том числе теоретический обзор области и системы, технологическое проектирование и описание системы, а также непосредственно результаты разработки.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Описание производственного процесса

На заводе «Шелезяка» осуществляется конвейерное производство печатных плат. Для этого завод оснащен тремя последовательно действующими информационно-управляющими системами. Рассмотрим этапы производства печатных плат на данном заводе. Первый этап представляет собой припайку необходимых компонентов на заготовку печатной платы. Здесь информационно-управляющая система осуществляет точное и автоматизированное нанесение компонентов на основу платы. На втором этапе готовая печатная плата упаковывается в специальную защитную упаковку. Этот этап важен для сохранения целостности и безопасности готовой продукции в процессе транспортировки и хранения. Третий этап предусматривает нанесение уникального кода на упаковку печатной платы. Информационно-управляющая система автоматически осуществляет эту операцию, обеспечивая уникальную идентификацию каждой платы. После выполнения третьего этапа готовые платы отгружаются на склад согласно маркировке, нанесенной на упаковке на предыдущем этапе производства.

1.2 Описание программируемой системы

Для автоматизации процесса используется информационно-управляющая система, состоящая из конвейерной ленты с электроприводом (Q0), контейнера с заготовками на начале ленты, заслонки которого открывается при помощи привода (Q1), и датчика (I1) для определения наличия заготовок в контейнере. Датчик (I0) на конце ленты сообщает о достижении заготовкой края ленты. На конце ленты располагается контейнер для готовых печатных плат с крышкой на электроприводе (Q2) и датчиком наполнения (I2). В середине ленты находится автоматический станок печати (Q3), перед которым располагается датчик

приближения заготовки (I3), обеспечивающий опускание оттиска и печать на заготовке. Пульт оператора содержит кнопку старта (I4), экстренной остановки (I5), а также индикаторы состояния: сигнализация об опустошении контейнера заготовок (Q6), заполненном контейнере для готовых печатных плат (Q7) и аварийном состоянии (Q8). Схематическое изображение установки (см. Рисунок 1.1).

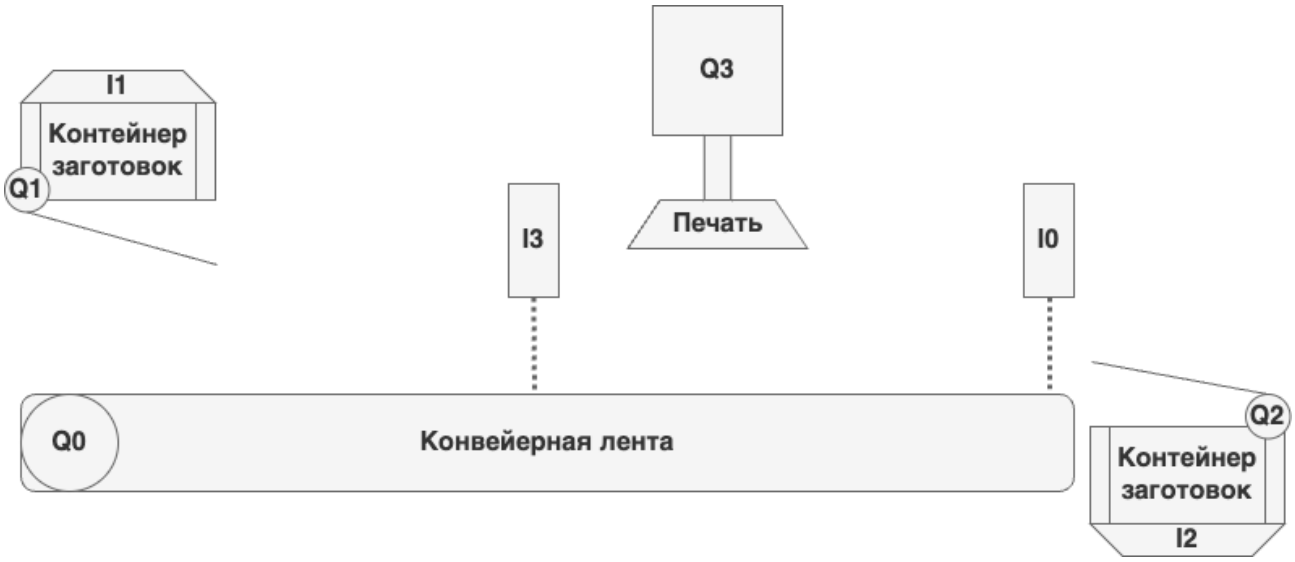

Рисунок 1.1 — Схематическое изображение установки

Этапы работы системы:

1. Оператор инициирует запуск ленты нажатием кнопки старта.

2. Выполняется проверка состояния контейнера с заготовками, с последующим выполнением соответствующих действий:

2.1.В случае обнаружения пустого контейнера с заготовками происходит вызов сигнала и остановка ленты.

2.2.Если контейнер с заготовками не является пустым, осуществляется его открытие.

3. Заготовка перемещается по ленте до момента достижения датчика печати.

4. После достижения заготовкой датчика печати производится опускание печати.

5. Деталь продолжает движение по ленте.

6. По достижении деталью конца ленты происходит открытие крышки контейнера для готовых заготовок.

7. Осуществляется проверка состояния контейнера с заготовками и выполнение соответствующих действий:

7.1.В случае заполнения контейнера происходит остановка процесса.

7.2.Если контейнер не заполнен, процесс продолжается с шага 2.

8. В любой момент, при нажатии кнопки остановки, процесс должен быть прекращен.

9. При переходе в неопределенные состояния процесс должен быть автоматически остановлен.

Для удобства исследования состояний работы информационно-управляющей системы представлена диаграмма состояний (см. Таблица 1.1). В данной диаграмме символ «X» используется для обозначения того, что состояние конкретного элемента системы (например, датчика) не является значимым в данном контексте.

Таблица 1.1 – Состояния информационно-управляющей системы

Шаг	**I0**	**I1**	**I2**	**I3**	**I4**	**I5**	**Q0**	**Q1**	**Q2**	**Q3**	**Q6**	**Q7**	**Q8**
1	0	X	0	0	1	0	1	0	0	0	0	0	0
2.1	0	0	0	0	0	0	0	0	0	0	1	0	1
2.2	0	1	0	0	0	0	1	1	0	0	0	0	0
3	0	X	0	0	0	0	1	0	0	0	0	0	0
4	0	X	0	1	0	0	1	0	0	1	0	0	0
5	0	X	0	0	0	0	1	0	0	0	0	0	0
6	1	X	0	0	0	0	1	0	0	0	0	0	0
7.1	0	X	1	0	0	0	0	0	1	0	0	1	1
7.2	0	X	0	0	0	0	1	0	0	0	0	0	0
8	X	X	X	X	X	1	0	0	0	0	0	0	0
9	X	X	X	X	X	X	0	0	0	0	0	0	1

1.3 Обзор существующих решений

С развитием промышленной автоматизации и использованием программируемых логических контроллеров (ПЛК) стало возможным решать широкий спектр задач в промышленных и производственных средах. Некоторые ПЛК-системы включают функционал для выполнения сложных операций, что является важным элементом для реализации различных управляющих и регулирующих алгоритмов. Программное обеспечение для ПЛК обычно включает в себя специальные инструкции или функции, которые позволяют работать с данными, осуществлять управление вводом-выводом и взаимодействовать с оборудованием. Это обеспечивает возможность автоматического выполнения операций непосредственно на уровне управления процессами в промышленной среде. Такие возможности помогают оптимизировать производственные процессы, упрощают разработку и поддержку управляющих систем, а также повышают эффективность работы оборудования в промышленных предприятиях. Вместе с другими функциями управления и анализа данных, выполнение операций на ПЛК позволяет создавать более гибкие и точные системы автоматизации.

Siemens SIMATIC PLC

Системы Siemens SIMATIC PLC (Программируемые логические контроллеры) широко признаны и используются в промышленной автоматизации за свою надежную работу и обширные возможности. Эти системы PLC предлагают множество функций, делая их незаменимыми в различных промышленных секторах.

Основные преимущества [1]:

- Широкий выбор моделей;
- Обширный функционал;
- Надежность и долговечность;
- Интегрированная среда разработки (IDE).

Основные недостатки [1]:

- Сложность для новичков;
- Высокие затраты;
- Ограниченная совместимость с другими системами.

Rockwell Automation Allen-Bradley PLC Systems

Системы Allen-Bradley PLC от Rockwell Automation широко известны и широко применяются в промышленности благодаря своей надежности, гибкости и мощности. Эти PLC системы предлагают широкий спектр функций и возможностей, делая их идеальным выбором для различных задач автоматизации.

Основные преимущества [2]:

- Широкий выбор моделей;
- Простота программирования;
- Надежность и долговечность;
- Обширная поддержка и экосистема.

Основные недостатки [2]:

- Высокие затраты;
- Ограниченная совместимость с другими системами.

PLC Systems

Компания ПЛКСистемы предлагает широкий спектр оборудования и комплексных решений для автоматизации промышленных процессов. Программируемые контроллеры XINJE XSF характеризуются ультракомпактным корпусом, поддержкой множества протоколов связи и гибкостью программирования.

Основные преимущества [3]:

- Ультракомпактные размеры;
- Поддержка множества протоколов связи;
- Гибкость и открытость в программировании;
- Простота отладки и сервисного обслуживания оборудования.

Основные недостатки [3]:

- Необходимость специализированных знаний и опыта для разработки и работы;
- Высокие затраты.

Emerson DeltaV Automation Platform

Платформа автоматизации DeltaV от Emerson предлагает современные технологические и программные решения для снижения рисков и повышения эффективности использования. Эта платформа характеризуется лёгкостью в использовании, обеспечивает повышенную гибкость и возможность масштабирования систем управления.

Основные преимущества [4]:

- Уменьшают сложность и риски проекта;
- Ускоряют процесс цифровой трансформации и модернизации;
- Повышают эффективность операций;
- Предоставляют контекстуализированные данные для принятия обоснованных управленческих решений;
- Способствуют оптимизированной и устойчивой работе предприятия.

Основные недостатки [4]:

- Могут быть более сложными в освоении для новичков в автоматизации;
- Необходима адаптация под специфику конкретного предприятия для полного раскрытия потенциала системы.

МикроДАТ ПЛК

Системы МикроДАТ ПЛК предлагают эффективные решения для автоматизации в сферах металлообработки, металлургии, горнодобычи, транспортировки нефти и газа и в агропромышленном комплексе. Компактные и гибкие в настройке, контроллеры МикроДАТ способны удовлетворить специфические требования различных промышленных процессов.

Основные преимущества [5]:

- Модульность и гибкость архитектуры;

- Расширенный диапазон рабочих температур;
- Высокая надежность;
- Высокое быстродействие и широкие коммуникационные возможности.

Основные недостатки [5]:

- Могут потребоваться специализированные знания для настройки и программирования;
- Инвестиционные затраты на модернизацию оборудования и обучение персонала;
- Необходимость адаптации системы под конкретные задачи предприятия.

1.4 Требования к программируемой системе

В таблице 1.2 представлены требования к программируемой системе.

Таблица 1.2 – Требования к программируемой системе

№	Требование	Значение
1	Язык программирования	C++
2	Корректность работы	Приложение запускается и поддерживает стабильный цикл работы от момента старта до завершения
3	Применение принципов объектно-ориентированного программирования	При написании приложения, как минимум, были использованы классы в C++, объектный подход к проектированию системы, а также инкапсуляция.
4	Интерфейс пользователя	Создан интерфейс пользователя, поддерживающий корректный пользовательский опыт и содержащий все необходимые пояснения к работе и эксплуатации
5	Инструкция по эксплуатации	Написана инструкция по эксплуатации, содержащая, в том числе, основные рекомендации по использованию и пояснения к возможным ошибкам в программе
6	Безопасность	Обеспечить защиту оператора и оборудования от неправильного использования и возможных аварийных ситуаций.
7	Адаптация к изменениям в производстве	Возможность быстрой адаптации и модификации системы в соответствии с изменениями в процессе производства и требованиями заказчика.
8	Поддержка нескольких языков интерфейса	Возможность выбора языка интерфейса предоставляется на русском, английском и китайском языках. Для установки предпочитаемого языка пользователь может обратиться к разделу "Настройки".
9	Мониторинг работы конвейерного оборудования	Создание системы контроля за функционированием конвейерной ленты, включающей в себя возможности мониторинга текущего состояния.
10	Поддержка различных операционных систем	Обеспечение совместимости программы с различными операционными системами (Windows, Linux, MacOS) для расширения круга потенциальных пользователей.

2 ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

2.1 Диаграмма состояний системы

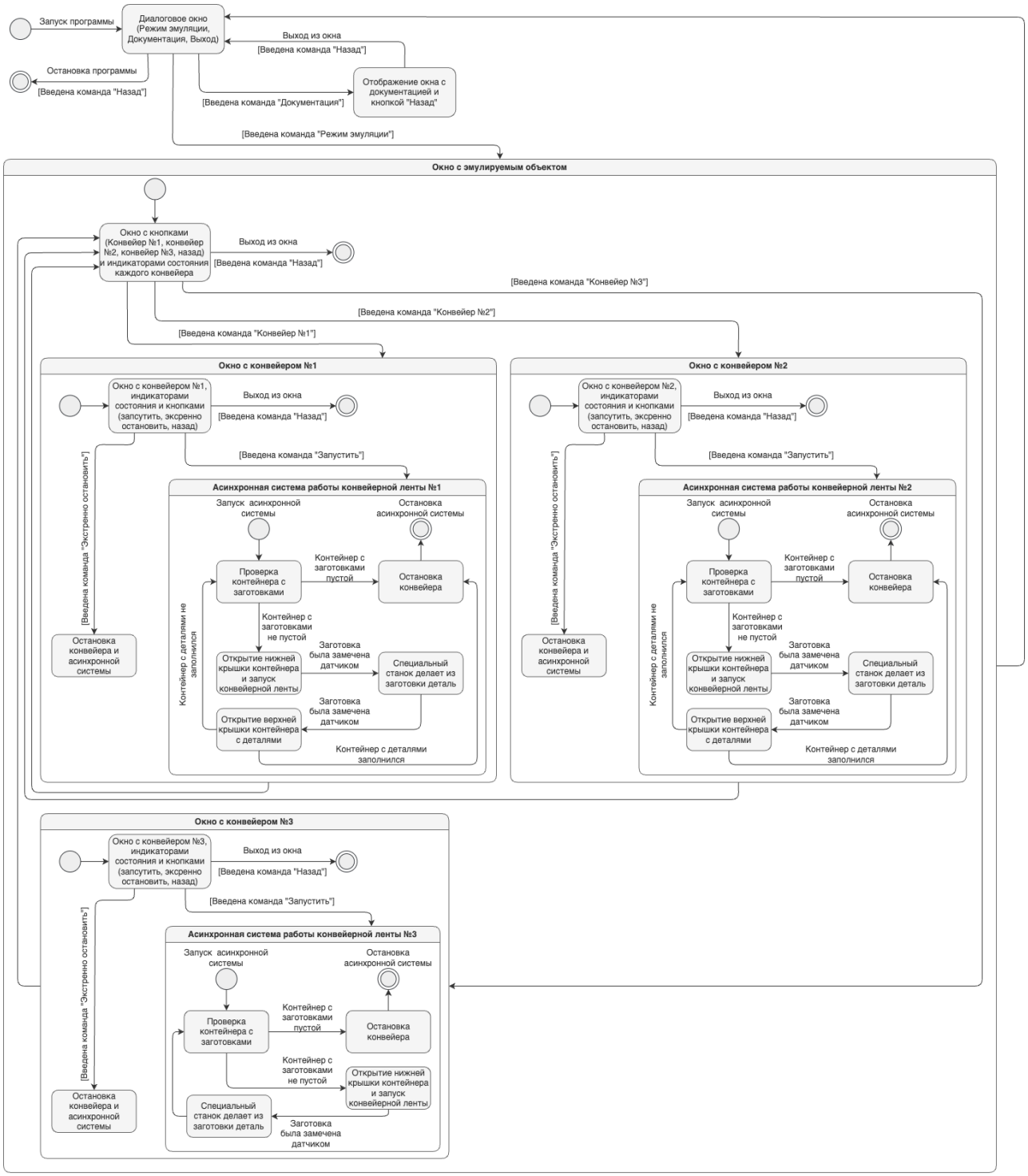

Рисунок 2.1 — Диаграмма состояний системы

На диаграмме, представленной на рисунке 2.1 показаны состояния программируемой системы. Давайте проанализируем внутреннюю структуру

более детально. После запуска программы пользователь встречается с диалоговым окном, содержащим меню с доступными опциями («Режим эмуляции», «Документация», «Выход»). При выборе опции «Выход» программа завершает свою работу. В случае выбора опции «Документация» пользователю предоставляется новый экран с соответствующей документацией и возможностью вернуться в главное меню. При выборе опции «Режим эмуляции» открывается новый экран с меню, где пользователь может выбрать доступные конвейерные ленты для управления. На данных экранах, отображено состояние конвейерной ленты и пульт управления. Для повышения удобства использования программы было решено разделить её на два асинхронных потока: первый поток отвечает за интерфейс, а второй – за эмуляцию информационно-управляющей системы. Асинхронная эмуляция информационно-управляющей системы производится по плану, написанном в описании программируемой системы.

2.2 Диаграмма классов системы

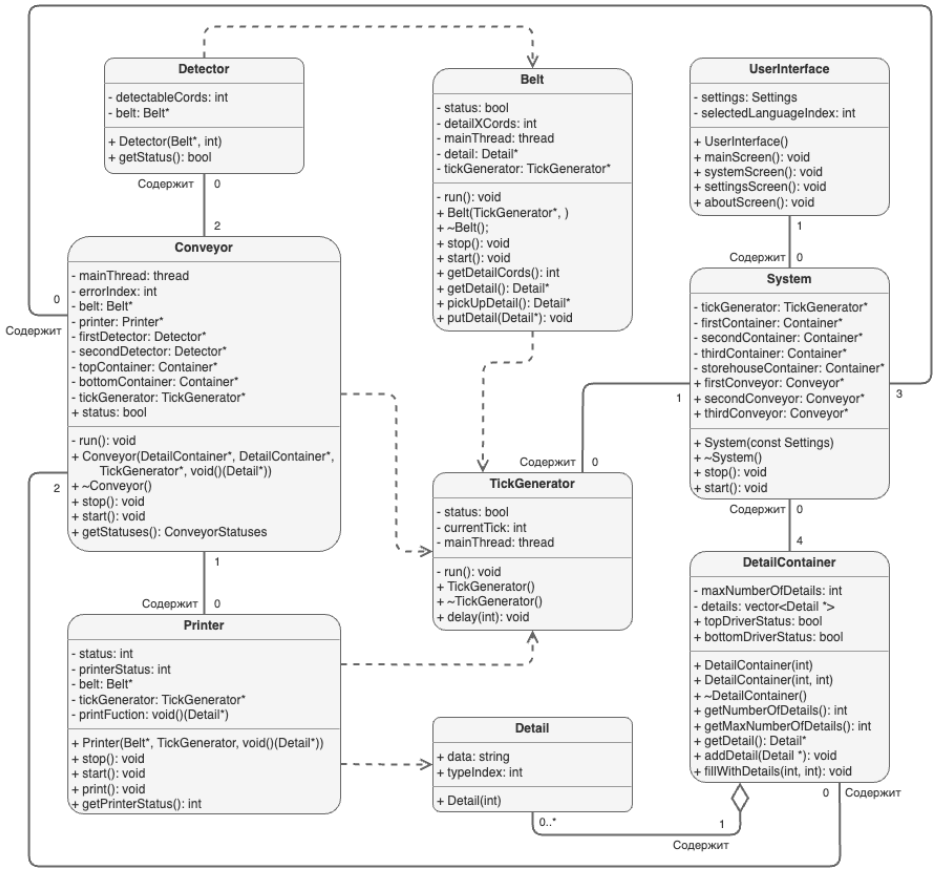

Рисунок 2.2 — Диаграмма классов системы

2.3 Диаграмма последовательности системы

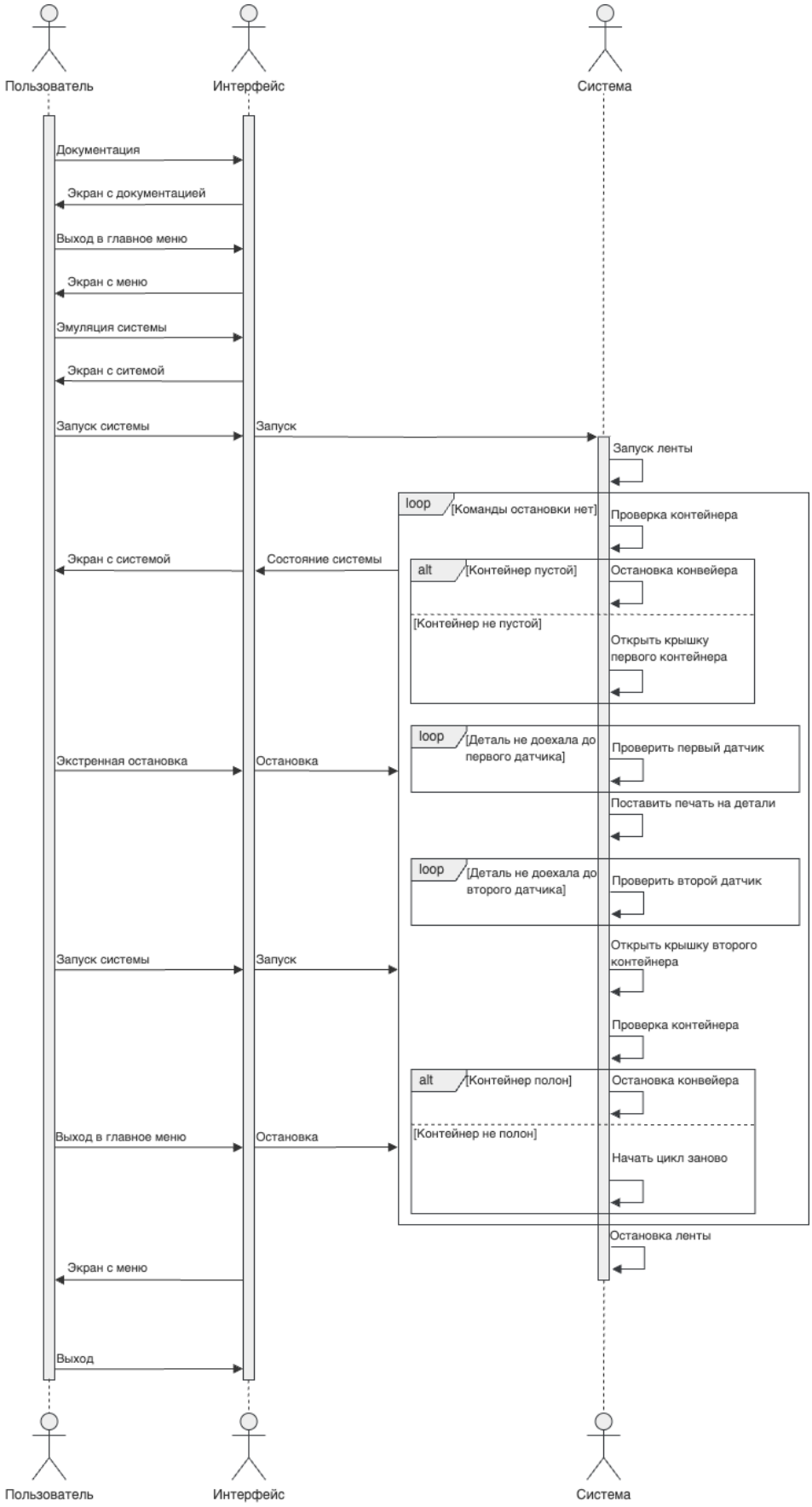

Рисунок 2.3 — Диаграмма последовательности системы

На диаграмме, изображенной на рисунке 2.3, представлена часть последовательности функционирования программной системы. Данная диаграмма визуализирует основную последовательность операций программы. В целях оптимизации размеров диаграммы было принято решение сократить повторяющиеся участки, такие как работа конвейерных лент. Кроме того, следует уточнить, что момент, когда интерфейс отображает текущее состояние системы, наступает непрерывно в реальном времени.

3 ПРАКТИЧЕСКАЯ ЧАСТЬ

3.1 Реализация требований к системе

Требование №1

Исходный код программного продукта полностью составлен на высокопроизводительном языке программирования C++, что обеспечивает стабильность и высокую скорость обработки данных. В подтверждение этого факта в приложении А документа листинги от А.1 до А.7 приведены листинги исходного кода, которые могут служить наглядным свидетельством использования языка C++ на всех этапах разработки.

Требование №2

Функциональность полного рабочего цикла приложения была подвергнута тщательному тестированию и успешной проверке квалифицированным преподавателем. Были проведены многочисленные испытания в лабораторных условиях, что позволило точно оценить производительность программного обеспечения и его способность эффективно функционировать в различных эксплуатационных сценариях, что полноценно подтверждает соответствие продукта заявленным требованиям.

Требование №3

Объектно-ориентированный подход является фундаментом сооружения данной программной системы, и наличие классов в исходном коде подчеркивает архитектурную стройность продукта. Эти аспекты находят свое отражение в приложении А листингах А.3 и А.8, включающих в себя соответствующие фрагменты кода, что поддерживает наше утверждение об использовании классов для достижения модульной структуры программы.

Требование №4

Важной составляющей привлекательности данного программного продукта является его многоязычный и интуитивно понятный интерфейс. Кропотливая работа над пользовательским интерфейсом позволила воплотить в жизнь функциональность, предоставляющую пользователю свободу выбора предпочитаемого языка из доступного списка, что делает использование программы комфортным для аудитории различных культурных и языковых групп.

Требование №5

Полноценное представление документации исходит из нашего стремления к предоставлению пользователю всех необходимых сведений для работы с программным продуктом. Раздел 3.3 прилагаемой документации содержит всеобъемлющее и понятное руководство пользователя, которое включает в себя пошаговые инструкции и рекомендации по обращению с оборудованием, облегчая внедрение и использование программы в повседневной деятельности конечного пользователя.

Требование №6

Особое внимание было уделено механизмам уведомления об ошибках и исключительных ситуациях в программе. Внедрена система отображения информативных сообщений об ошибке, которая активизируется при возникновении нежелательных или непредвиденных обстоятельств в ходе функционирования приложения. Данный механизм помогает в оперативном решении проблем и минимизации времени простоя системы. Как пример, на Рисунке 3.1 изложена ошибочная ситуация, сигнализирующая о недостаче запчастей, что предотвращает продолжение процесса работы конвейера, иллюстрируя высокую информативность и прозрачность системы отчетности программного продукта.

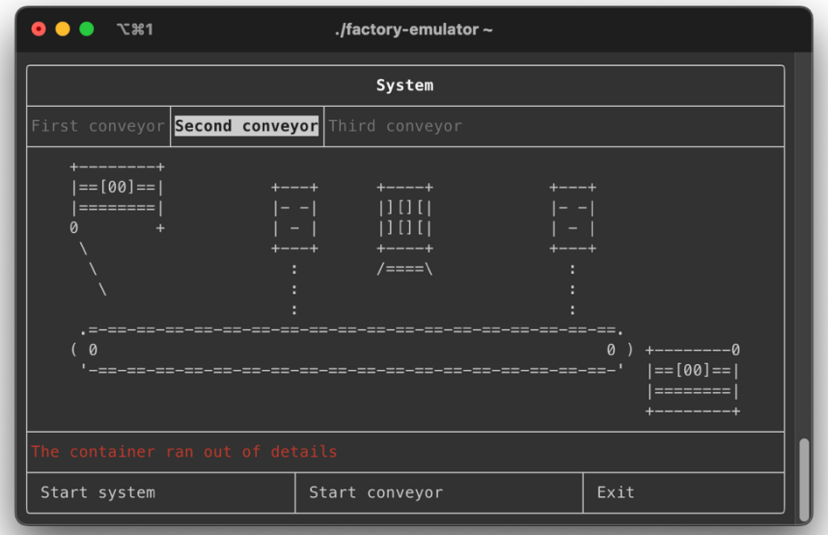

Рисунок 3.1 — Пример отображения ошибок в программе

Требование №7

В разделе «Настройки» пользователь имеет возможность тонкой настройки количества исходных компонентов, содержащихся в контейнерах, что позволяет точно адаптировать программу под индивидуальные производственные процессы и уникальные требования каждого предприятия. Эта гибкость настройки исключительно полезна для максимальной оптимизации ресурсов и повышения эффективности рабочих процессов (см. Рисунок 3.2).

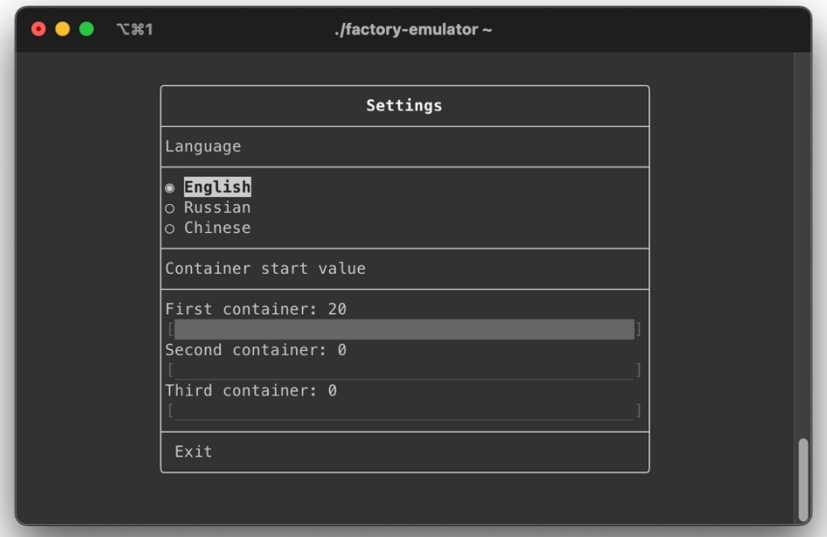

Рисунок 3.2 — Меню «Настройки» на английском языке

Требование №8

Во время разработки программы было уделено особое внимание удобству использования продукта аудиторией из различных уголков планеты. В разделе «Настройки» реализована поддержка нескольких наиболее используемых языков, что делает программу максимально инклюзивной и толерантной к языковым переплетениям современного мультикультурного мира. Благодаря этому, практически любой пользователь, независимо от своего местоположения, может настроить интерфейс на понятный ему язык, что существенно облегчает взаимодействие с программой (соответственно см. Рисунок 3.2, Рисунок 3.3, Рисунок 3.4).

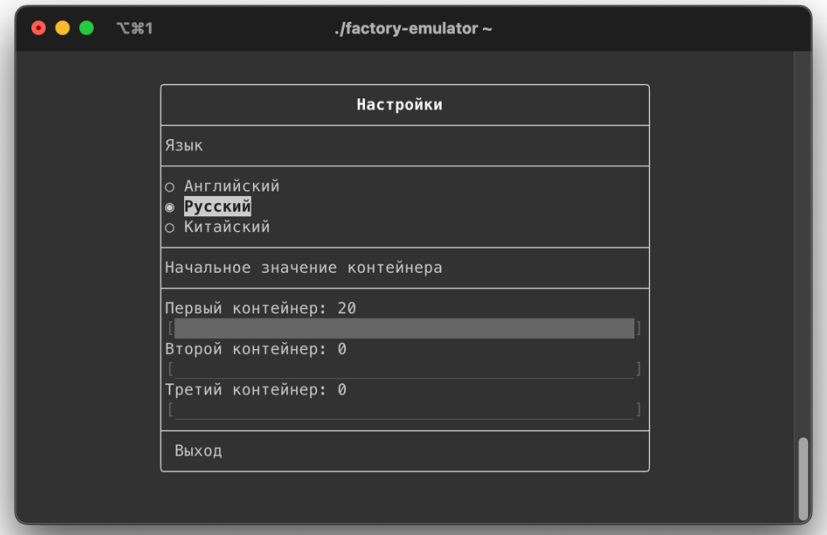

Рисунок 3.3 — Меню «Настройки» на русском языке

Рисунок 3.4 — Меню «Настройки» на китайском языке

Требование №9

В рамках разработки программного обеспечения была внедрена функциональность динамического экрана, который в реальном времени визуализирует текущее состояние работы конвейерной системы. Это позволяет пользователю наглядно отслеживать и оценивать процессы, происходящие в производственном потоке, обеспечивая тем самым повышенную надёжность контроля за операциями на конвейере (для детального изучения см. Рисунок 3.5).

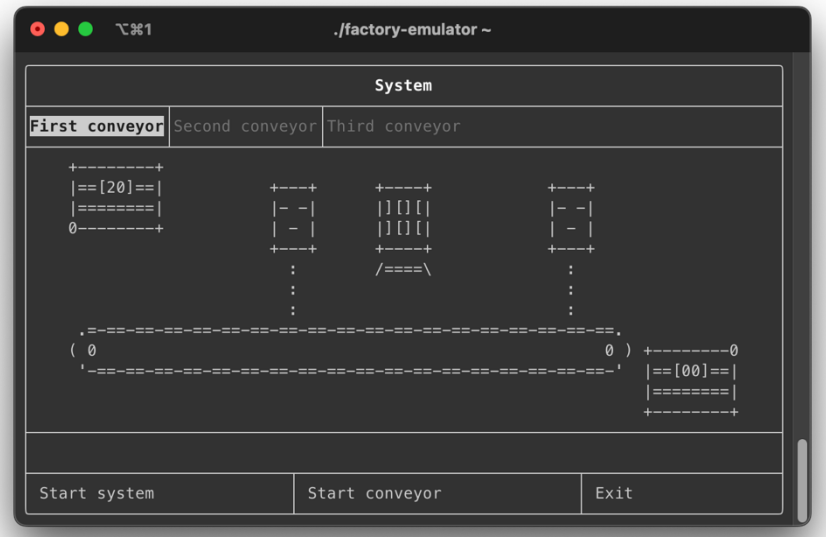

Рисунок 3.5 — Меню «Система» в MacOS

Требование №10

В ходе разработки данного программного продукта особое внимание было уделено кроссплатформенной совместимости, что достигнуто благодаря интеграции библиотек, имеющих поддержку в широком спектре операционных систем. Это породило возможность компиляции и адаптации программного обеспечения для эксплуатации в разнообразных средах пользователя, включая, но не ограничиваясь операционными системами MacOS, Linux и Windows. Демонстрация стабильной работы приложения в указанных ОС представлена на Рисунках 3.5, 3.6 и 3.7 соответственно. Такой подход в разработке позволяет обеспечить высокую универсальность и доступность программного продукта для широкого круга пользователей.

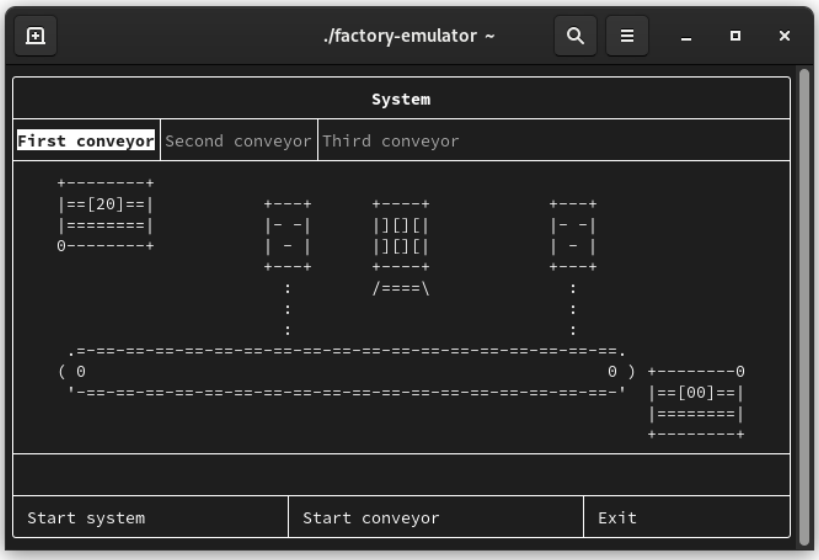

Рисунок 3.6 — Меню «Система» в Rocky Linux

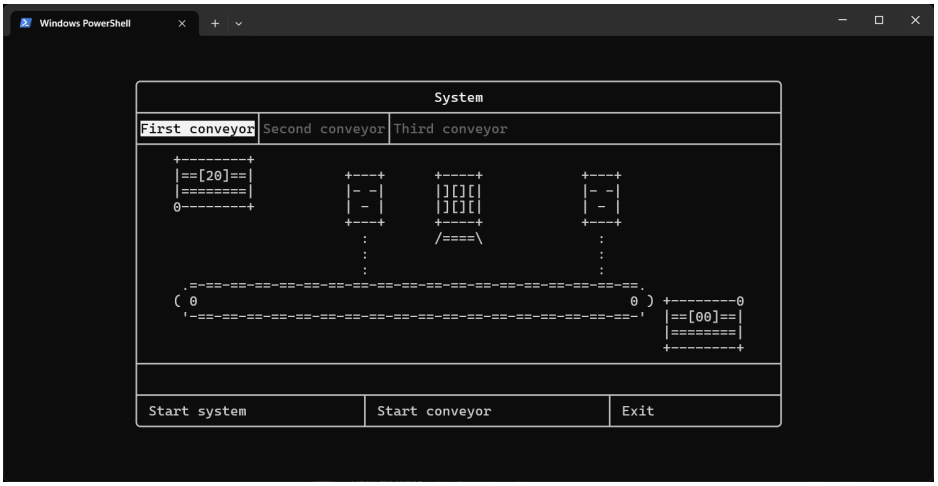

Рисунок 3.7 — Меню «Система» в Windows 11

3.2 Функциональное тестирование программного продукта

Начальный этап работы с нашей программой характеризуется представлением главного меню пользователю. Это важнейшая часть интерфейса, призванная обеспечить доступ к различным функциональным возможностям программы в удобной и структурированной форме (см. Рисунок 3.8).

Рисунок 3.8 — Главное меню программы

Каждый пункт меню был разработан с использованием отдельной функции, что уходит корни в наших усилиях по обеспечению модульности и расширяемости. Это подход обеспечивает гибкость и способствует простоте обслуживания и возможностям обновления нашего программного обеспечения. Главная функция программы реализована таким образом, чтобы обеспечивать координированную работу модулей, отвечающих за взаимодействие с интерфейсом, и модулей, обеспечивающих работу отдельных экранов приложения. Вместе они составляют интерфейс пользователя, ставя в центр внимания удобство и интуитивность использования (см. Приложение А, Листинг А.2 и Листинг А.3).

Меню «Настройки» представляет собой ключевой элемент программной архитектуры, который предлагает пользователям инструменты для фундаментальной настройки системных параметров. Через этот модуль доступна

опция конфигурации начального количества компонентов в каждом контейнере, что позволяет оптимизировать и адаптировать рабочий процесс к специфическим требованиям (см. рисунок 3.2). Кроме того, данный раздел содержит настройки локализации, предоставляя возможность изменения языка интерфейса в соответствии с предпочтениями пользователя, что обеспечивает более комфортное взаимодействие с программой для неанглоязычных пользователей. Функциональные возможности и детали реализации экрана «Настройки» изложены в Приложении А, включающем листинг кода А.2, где можно ознакомиться с полным набором инструкций и протоколов, задействованных в этом процессе.

В контексте интерфейса пользователя системы моделирования промышленных процессов, раздел «Система» задуман как центральное место для инициализации и мониторинга операций условного заводского производства. Данный интерфейс представляет собой сложное многофункциональное пространство, содержащее унифицированные инструменты для визуализации и координации работы конвейерных лент. Используя расположенные в верхней части экрана вкладки, пользователь имеет возможность с легкостью переключаться между различными этапами производственного цикла, что способствует оперативной и эффективной организации рабочего процесса. Каждая вкладка в интерфейсе является своеобразным порталом в соответствующий сегмент системы, гарантируя таким образом максимально быстрый и удобный доступ к необходимым функциям. В нижней части экрана, для обеспечения компетентного управления как системой в целом, так и ее индивидуальными составляющими, реализован набор контрольных кнопок. Эти элементы управления дополнены интуитивно понятными подписями, что обеспечивает наглядность и удобство в использовании. Для подробного ознакомления с дизайном и функционалом управляющего интерфейса предлагается обратиться к рисунку 3.5, где демонстрируются ключевые аспекты управленческой платформы. На экране управления системы моделирования промышленных процессов предусмотрен специализированный раздел, цель

которого – отображение сведений о возможных дисфункциях или неисправностях в работе отдельных конвейерных лент. Данное решение обеспечивает оперативное информирование оператора о технических отклонениях, что играет важнейшую роль в поддержании непрерывности производственного процесса. Благодаря реализации данной функции, персонал управления имеет возможность своевременно диагностировать и производить корректировку параметров системы, тем самым минимизируя время простоя и предотвращая потенциальные сбои в производственном цикле. Элементы индикации ошибок спроектированы с применением высоких стандартов визуализации, что делает процесс мониторинга интуитивно понятным и удобным для персонала. Для иллюстрации принципа работы и визуального представления указанного раздела, предлагается ознакомиться с рисунком 3.1, который детализирует интерфейс данного сегмента управляющей системы и демонстрирует взаимодействие между индикаторами системы и пользовательскими действиями в контексте предупреждения и реагирования на экстренные производственные обстоятельства.

В процессе проектирования системы моделирования операций на производственных мощностях, основное внимание уделяется структурированию и внедрению набора изощренных классов, каждый из которых занимается конкретным сегментом в механизме конвейерной ленты. Этот подход содействует созданию солидной основы для исчерпывающей модульности и упрощения процедур дальнейших модификаций и расширений системы. Спецификация и последовательность операций, выполняемых конвейерной лентой, неукоснительно согласуются с методическими указаниями, предоставленными в разделе 1.2. Рассматриваемый программный код интегрирует описанные принципы и механизмы работы, воплощая их в определенной манере. Имплементация каждого класса включает в себя эксклюзивный ассортимент свойств и методов, которые служат для достижения высокой градации ответственности и дисциплинированных взаимодействий между различными частями конвейера. Более того, такой дизайн облегчает

обслуживание кода и дает возможность детального контроля над его поведением. Для того, чтобы познакомиться с техническими деталями реализации, обращение к Приложению А предоставит исчерпывающие примеры (см. Листинги А.7 и А.8), где подробно описывается комбинирование различных элементов кода с целью создания полноценного и функционального модуля. Это, в свою очередь, способствует укреплению архитектурной целостности программного обеспечения и гарантирует его надежное функционирование в широком диапазоне производственных сценариев.

3.3 Инструкция по эксплуатации

При инициализации программного обеспечения пользователь немедленно попадает в стартовое меню, представляющее из себя прекратно структурированный пользовательский интерфейс. Для навигации между различными функциональными разделами и подменю заложены два основных метода управления: перемещение осуществляется благодаря использованию кнопок-стрелок на клавиатуре, предназначенных для курсорной навигации, либо с помощью наведения координатного указателя мыши на интересующий элемент управления. Выбор конкретного элемента в процессе работы с программой в главном меню подразумевает нажатие клавиши Enter, что служит актом подтверждения осуществленного выбора, и убедительно исключает вероятность ошибочного срабатывания действий в случайном порядке. Для подробного ознакомления с процедурой перемещения по меню и механизмом подтверждения выбора рекомендуется рассмотреть графическое представление, представленное на рисунке 3.8. Дизайн интерфейса и схема навигации разработаны таким образом, чтобы максимально облегчить процедуру взаимодействия пользователя с программным продуктом, делая процесс выбора интуитивно понятным и минимизируя время на ознакомление с функционалом.

На экране «Настройки» представлены широкие возможности для гибкой конфигурации системных параметров программного продукта. В частности,

здесь можно произвести критически важные действия, такие как корректировка языка интерфейса для обеспечения комфортного взаимодействия пользователя с программой в его предпочитаемом лингвистическом формате. Кроме того, доступна функция установки начальных значений для контейнеров данных, что является ключевым шагом в предварительной подготовке приложения к эффективной эксплуатации. Выбор параметров на данном экране реализован аналогичным образом, как и в других разделах приложения: посредством курсорной навигации с помощью стрелок на клавиатуре или через прямое взаимодействие указателем мыши, что подразумевает общность и универсальность методов управления на протяжении работы с программой. Для детального представления о механизме настройки и выбора параметров рекомендуется обратить внимание на информацию, изображенную на рисунке 3.2.

В структуре интерфейса меню системы предусмотрена возможность навигации пользователя по ключевым разделам приложения, которые размещены в верхней части экрана в форме маркированных вкладок. Это позволяет обеспечить целенаправленный и быстрый доступ к необходимым функциональным аспектам управления. Осуществление контроля за процессами, проходящими на производственных линиях и отдельно взятых конвейерных агрегатах, возлагается на комплекс манипуляционных элементов, корректно отобранных и расположенных в нижней части экрана (см. Рисунок 3.5). При возникновении дисфункций в производственном процессе в центральной части экрана генерируется и отображается сообщение об ошибке, пример чего иллюстрируется на рисунке 3.1. Наиболее распространенными типами таких сообщений являются указания на недостаток или избыток заготовок в специализированных контейнерах. Для успешной корректировки этой категории ошибок, управляющему предлагается опция разгрузки или регулировки содержимого контейнеров через инструментарий, локализованный в меню настроек.

ЗАКЛЮЧЕНИЕ

В ходе работы была произведена тщательная разработка программно-аппаратной комплексной системы, произведен анализ и сопоставление с аналогами, действующими на рынке. Основываясь на глубоком изучении предметной области, были выработаны ключевые требования к проектируемому решению. Следуя этим требованиям, был создан проект структуры системы с включением диаграмм состояний, диаграмм взаимодействия классов и последовательности процессов внутри системы. Разработанный программный продукт был реализован с соблюдением всех поставленных задач, предварительно протестирован для проверки соответствия спецификациям и функциональным ожиданиям. Дополнительно была оформлена подробная инструкция по эксплуатации разработанного программного обеспечения, обеспечивающая необходимую поддержку пользователям при работе с системой.

Дальнейшие перспективы эволюции и адаптации разработанного программного решения предполагают его стратегическое внедрение в структурные подразделения корпоративного сектора в качестве инструментария для оптимизации управленческих процессов и повышения эффективности контроля производственной деятельности. Программный продукт обладает значительным потенциалом для интеграции и автоматизации ключевых аспектов технологической и административной инфраструктуры предприятий, что позволит достичь более высоких показателей управляемости, гибкости процессов и реагирования на динамически меняющиеся рыночные условия. Внедрение такового ПО сопровождается не только с целью наращивания производительности, но и с учетом стратегии непрерывного совершенствования качества и функциональности системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Siemens S7-1500 Программируемый контроллер / [Электронный ресурс] // Siemens : [сайт]. — URL: https://www.siemens-pro.ru/components/s7-1500.htm (дата обращения: 09.03.2024).

2. PLC Programmable Controllers / [Электронный ресурс] // Rockwell Automation : [сайт]. — URL: https://www.rockwellautomation.com/en-us/products/hardware/allen-bradley/programmable-controllers.html (дата обращения: 09.03.2024).

3. ПЛКСистемы / [Электронный ресурс] // PLC Systems : [сайт]. — URL: https://www.plcsystems.ru/ (дата обращения: 09.03.2024).

4. Emerson DeltaV Automation Platform / [Электронный ресурс] // Emerson : [сайт]. — URL: https://www.emerson.com/en-us/automation/deltav (дата обращения: 09.03.2024).

5. МикроДАТ / [Электронный ресурс] // МикроДАТ : [сайт]. — URL: https://www.microdat.ru/ (дата обращения: 11.03.2024).

ПРИЛОЖЕНИЯ

Приложение А – Исходный код программы

Листинг А.1 — Файл src/envoironment.h

```
#ifndef environment
#define environment

#include <string>

using namespace std;

const string projectCreator = "NKTKLN";
const string projectCodename = "Gamma";

#endif
```

Листинг А.2 — Файл src/interface.cpp

```
#include <string>
#include <vector>

#include <fmt/core.h>

#include "ftxui/dom/elements.hpp"
#include "ftxui/screen/color.hpp"
#include "ftxui/screen/terminal.hpp"
#include "ftxui/component/component.hpp"
#include "ftxui/component/captured_mouse.hpp"
#include "ftxui/component/component_options.hpp"
#include "ftxui/component/screen_interactive.hpp"

#include "models.h"
#include "system.h"
#include "interface.h"
#include "environment.h"

using namespace std;
using namespace fmt;
using namespace ftxui;

UserInterface::UserInterface() : selectedLanguageIndex(0) {
    settings.containerMaxValue = 20;
    settings.firstContainerValue = 20;
    settings.secondContainerValue = 0;
    settings.thirdContainerValue = 0;
    settings.language = languages[selectedLanguageIndex];
}

void UserInterface::mainScreen() {
    if (Terminal::Size().dimy < 23 || Terminal::Size().dimx < 80) {
        Element document = color(Color::Red, hbox({
            text("Terminal window size is less than "),
            text("80x23") | underlined,
            text(" characters!"),
        })) | bold;
```

```
        Screen screen = Screen::Create(Dimension::Full(),
Dimension::Fit(document));
        Render(screen, document);
        screen.Print();
        return;
    }

    ScreenInteractive screen = ScreenInteractive::Fullscreen();

    int selected = 0;
    MenuOption option = MenuOption();
    option.on_enter = [&] {
        switch (selected) {
        case 0: systemScreen(); break;
        case 1: settingsScreen(); break;
        case 2: aboutScreen(); break;
        case 3: screen.Exit(); break;
        }
    };
    Component menu = Menu(&settings.language.mainMenuEntries, &selected,
option);
    Component renderer = Renderer(menu, [&] {
        return vbox({
            text(settings.language.mainMenuTitle) | center | bold,
            separator(),
            menu->Render() | size(WIDTH, EQUAL, 20),
        }) | border | center;
    });

    screen.Loop(renderer);
}

void UserInterface::systemScreen() {
    ScreenInteractive screen = ScreenInteractive::Fullscreen();
    atomic<bool> refresh_ui_continue = true;
    thread refresh_ui([&] {
        while (refresh_ui_continue) {
            this_thread::sleep_for(0.05s);
            screen.Post(Event::Custom);
        }
    });

    System *mainSystem = new System(settings);
    Conveyor *conveyor = mainSystem->firstConveyor;

    Component exitButton = Button(&settings.language.exit, [&] { screen.Exit();
},
        ButtonOption::Ascii());

    bool isSystemStarted = false;
    string systemButtonLable = settings.language.systemMenuStartButton;
    Component systemButton = Button(&systemButtonLable,
        [&] {
            if (!isSystemStarted) {
                mainSystem->start();
                systemButtonLable = settings.language.systemMenuStopButton;
            }
            else mainSystem->stop();
            isSystemStarted = !isSystemStarted;
        }, ButtonOption::Ascii());

    string conveyorButtonLable =
settings.language.systemMenuConveyorStartButton;
    Component conveyorButton = Button(&conveyorButtonLable,
```

```
        [&] {
            if (conveyor->status) conveyor->stop();
            else conveyor->start();
        }, ButtonOption::Ascii());

    int tabSelected = 0;
    Component tabToggle = Toggle(&settings.language.systemMenuTabs,
&tabSelected);

    Component buttons = Container::Horizontal({
        systemButton | flex,
        Renderer([&] { return separator(); }),
        conveyorButton | flex,
        Renderer([&] { return separator(); }),
        exitButton | flex
    });

    Component container = Container::Vertical({
        tabToggle,
        buttons
    });
    Component renderer = Renderer(container, [&] {
        switch (tabSelected) {
        case 0: conveyor = mainSystem->firstConveyor; break;
        case 1: conveyor = mainSystem->secondConveyor; break;
        case 2: conveyor = mainSystem->thirdConveyor; break;
        }

        if (!mainSystem->firstConveyor->status ||
            !mainSystem->secondConveyor->status ||
            !mainSystem->thirdConveyor->status) {
            isSystemStarted = false;
            systemButtonLable = settings.language.systemMenuStartButton;
        }
        else {
            isSystemStarted = true;
            systemButtonLable = settings.language.systemMenuStopButton;
        }

        conveyorButtonLable = conveyor->status ?
settings.language.systemMenuConveyorStopButton :
            settings.language.systemMenuConveyorStartButton;

        // Conveyor animation builder
        Component conveyorContainer = Container::Vertical({});
        ConveyorStatuses statuses = conveyor->getStatuses();
        int beltIndex = statuses.detailXCords >= 0 ? statuses.detailXCords % 3 :
0;
        for (int index = 0; index < 13; index++) {
            string line = vformat(conveyorTemplate[index],
                make_format_args(
                    containerStates[0][statuses.topContainer],
                    containerStates[1][statuses.topContainer],
                    statuses.topContainerValue,
                    containerStates[0][statuses.bottomContainer],
                    containerStates[1][statuses.bottomContainer],
                    statuses.bottomContainerValue,
                    beltStates[2 - beltIndex],
                    beltStates[beltIndex],
                    detectorStates[statuses.firstDetector],
                    detectorStates[statuses.secondDetector]
                ));

            if (index >= 6 && index <= 7 && statuses.detailType >= 0) {
```

```
                string detail = vformat(boxTypes[statuses.detailType][index -
6],
                    make_format_args(statuses.detailData));
                for (int pixelCord = 0; pixelCord < detail.size(); pixelCord++)
                    line[7 + pixelCord + statuses.detailXCords] =
detail[pixelCord];
            }

            if (index >= 5 && index <= 7) {
                string printer = (index - 5) == statuses.printer ?
printerStates[1] : printerStates[0];
                if (statuses.printer == 0 && index != 5 || (index - 5) >
statuses.printer) printer = "";
                for (int pixelCord = 0; pixelCord < printer.length();
pixelCord++)
                    line[32 + pixelCord] = printer[pixelCord];
            }

            conveyorContainer->Add(Renderer([line] { return text(line); }));
        }

        return vbox({
            text(settings.language.systemMenuTitle) | bold | center,
            separator(),
            tabToggle->Render(),
            separator(),
            conveyorContainer->Render() | center,
            separator(),
            text(settings.language.systemErrors[statuses.errorIndex]) |
color(Color::Red),
            separator(),
            buttons->Render(),
        }) | size(WIDTH, EQUAL, 78) | border | center;
    });
    screen.Loop(renderer);
    refresh_ui_continue = false;
    refresh_ui.join();
    delete mainSystem;
}

void UserInterface::settingsScreen() {
    ScreenInteractive screen = ScreenInteractive::Fullscreen();

    RadioboxOption option = RadioboxOption();
    option.focused_entry = selectedLanguageIndex;
    option.on_change = [&] { settings.language =
languages[selectedLanguageIndex]; };
    Component languageRadiobox =
Radiobox(&settings.language.settingsMenuLanguageEntries,
        &selectedLanguageIndex, option);
    Component exitButton = Button(&settings.language.exit, [&] { screen.Exit();
},
        ButtonOption::Ascii());
    Component firstContainerValueSlider = Slider("",
&settings.firstContainerValue,
        0, settings.containerMaxValue, 1);
    Component secondContainerValueSlider = Slider("",
&settings.secondContainerValue,
        0, settings.containerMaxValue, 1);
    Component thirdContainerValueSlider = Slider("",
&settings.thirdContainerValue,
        0, settings.containerMaxValue, 1);

    Component container = ftxui::Container::Vertical({
```

```
        languageRadiobox,
        firstContainerValueSlider,
        secondContainerValueSlider,
        thirdContainerValueSlider,
        exitButton,
    });
    Component renderer = Renderer(container, [&] {
        return vbox({
            text(settings.language.settingsMenuTitle) | bold | center,
            separator(),
            text(settings.language.settingsMenuLanguage),
            separator(),
            languageRadiobox->Render(),
            separator(),
            text(settings.language.settingsMenuConatinerValue),
            separator(),
            hbox(text(settings.language.settingsMenuFirstConatiner),
                 text(": " + to_string(settings.firstContainerValue))),
            firstContainerValueSlider->Render(),
            hbox(text(settings.language.settingsMenuSecondConatiner),
                 text(": " + to_string(settings.secondContainerValue))),
            secondContainerValueSlider->Render(),
            hbox(text(settings.language.settingsMenuThirdConatiner),
                 text(": " + to_string(settings.thirdContainerValue))),
            thirdContainerValueSlider->Render(),
            separator(),
            exitButton->Render(),
        }) | size(WIDTH, EQUAL, 50) | border | center;
    });
    screen.Loop(renderer);
}

void UserInterface::aboutScreen() {
    ScreenInteractive screen = ScreenInteractive::Fullscreen();
    Component exitButton = Button(settings.language.exit, [&] { screen.Exit();
},
        ButtonOption::Ascii());
    auto renderer = Renderer(exitButton, [&] {
        return vbox({
            text(settings.language.aboutMenuTitle) | bold | center,
            separator(),
            hbox(text(settings.language.aboutMenuCodename + ": "),
                color(Color::Green, text(projectCodename))),
            hbox(text(settings.language.aboutMenuCreator + ": "),
                color(Color::Cyan, text(projectCreator))),
            separator(),
            exitButton->Render(),
        }) | size(WIDTH, EQUAL, 20) | border | center;
    });
    screen.Loop(renderer);
}
```

Листинг А.3 — Файл src/interface.h

```
#ifndef interface
#define interface

#include "models.h"

class UserInterface {
private:
    Settings settings;
    int selectedLanguageIndex;

public:
```

```
    UserInterface();

    void mainScreen();
    void systemScreen();
    void settingsScreen();
    void aboutScreen();
};

#endif
```

Листинг А.4 — Файл src/main.cpp

```
#include "interface.h"

int main() {
    UserInterface ui = UserInterface();
    ui.mainScreen();
    return 0;
}
```

Листинг А.5 — Файл src/models.cpp

```
#include <string>

#include "models.h"

using namespace std;

// Interface language templates
Language English() {
    Language lang;

    lang.exit = "Exit";

    lang.mainMenuTitle = "Factory emulator";
    lang.mainMenuEntries = {
        "Start",
        "Settings",
        "About",
        lang.exit
    };

    lang.systemMenuTitle = "System";
    lang.systemMenuStopButton = "Stop system";
    lang.systemMenuStartButton = "Start system";
    lang.systemMenuConveyorStopButton = "Stop conveyor";
    lang.systemMenuConveyorStartButton = "Start conveyor";
    lang.systemErrors = {
        "",
        "The container ran out of details",
        "The maximum level of details in the container has been exceeded",
    };
    lang.systemMenuTabs = {
        "First conveyor",
        "Second conveyor",
        "Third conveyor"
    };

    lang.settingsMenuTitle = "Settings";
    lang.settingsMenuLanguage = "Language";
    lang.settingsMenuConatinerValue = "Container start value";
    lang.settingsMenuFirstConatiner = "First container";
    lang.settingsMenuSecondConatiner = "Second container";
    lang.settingsMenuThirdConatiner = "Third container";
```

```
    lang.settingsMenuLanguageEntries = {
        "English",
        "Russian",
        "Chinese"
    };

    lang.aboutMenuTitle = "About";
    lang.aboutMenuCodename = "Codename";
    lang.aboutMenuCreator = "Creator";

    return lang;
}

Language Russian() {
    Language lang;

    lang.exit = "Выход";

    lang.mainMenuTitle = "Эмулятор фабрики";
    lang.mainMenuEntries = {
        "Запуск",
        "Настройки",
        "О программе",
        lang.exit
    };

    lang.systemMenuTitle = "Система";
    lang.systemMenuStopButton = "Остановить систему";
    lang.systemMenuStartButton = "Запустить систему";
    lang.systemMenuConveyorStopButton = "Остановить конвейер";
    lang.systemMenuConveyorStartButton = "Запустить конвейер";
    lang.systemErrors = {
        "",
        "В корнейтене закончились детали",
        "Достигнут максимальный уровень деталей в контейнере",
    };
    lang.systemMenuTabs = {
        "Первый конвейер",
        "Второй конвейер",
        "Третий конвейер"
    };

    lang.settingsMenuTitle = "Настройки";
    lang.settingsMenuLanguage = "Язык";
    lang.settingsMenuConatinerValue = "Начальное значение контейнера";
    lang.settingsMenuFirstConatiner = "Первый контейнер";
    lang.settingsMenuSecondConatiner = "Второй контейнер";
    lang.settingsMenuThirdConatiner = "Третий контейнер";
    lang.settingsMenuLanguageEntries = {
        "Английский",
        "Русский",
        "Китайский"
    };

    lang.aboutMenuTitle = "О программе";
    lang.aboutMenuCodename = "Кодовое имя";
    lang.aboutMenuCreator = "Создатель";

    return lang;
}

Language Chinese() {
    Language lang;
```

```
    lang.exit = "退出";

    lang.mainMenuTitle = "工厂模拟器";
    lang.systemMenuStopButton = "停止系统";
    lang.systemMenuStartButton = "启动系统";
    lang.systemMenuConveyorStopButton = "停止传送带";
    lang.systemMenuConveyorStartButton = "启动传送带";
    lang.mainMenuEntries = {
        "启动",
        "设置",
        "关于",
        lang.exit
    };

    lang.systemMenuTitle = "系统";
    lang.systemMenuTabs = {
        "第一传送带",
        "第二传送带",
        "第三传送带"
    };

    lang.settingsMenuTitle = "设置";
    lang.settingsMenuLanguage = "语言";
    lang.settingsMenuConatinerValue = "容器初始值";
    lang.settingsMenuFirstConatiner = "第一容器";
    lang.settingsMenuSecondConatiner = "第二容器";
    lang.settingsMenuThirdConatiner = "第三容器";
    lang.systemErrors = {
        "",
        "没有关于集装箱的更多细节",
        "已超出容器中详细信息的最大级别",
    };
    lang.settingsMenuLanguageEntries = {
        "英语",
        "俄语",
        "汉语"
    };

    lang.aboutMenuTitle = "关于";
    lang.aboutMenuCodename = "代号";
    lang.aboutMenuCreator = "创建者";

    return lang;
}

extern const vector<Language> languages = {English(), Russian(), Chinese()};

// Conveyor screen templates

/*
0 - top box close
1 - top box open
```

```
2 - top box number
3 - second box close
4 - second box open
5 - second box number
6 - belt top
7 - belt bottom
8 - first detector
9 - second detector
*/
extern const string conveyorTemplate[13] = {
    "+--------+                                                                  ",
    "|==[{2:02}]==|           +---+      +-----+            +---+
",
    "|========|           |{8} {8}|      |][][|            |{9} {9}|
",
    "0{0}+           | - |      |][][|            | - |                ",
    " {1}                    +---+      +-----+            +---+                ",
    "  {1}                     :                             :                  ",
    "   {1}                    :                             :             {4}
",
    "                          :                             :               {4}  ",
    " .{6}.          {4} ",
    "( 0                                                        0 ) +{3}0",
    " '{7}'  |==[{5:02}]==|",
    "                                                             |========|",
    "                                                             +--------+"
};

extern const string containerStates[2][2] = {
    {"--------", "        "},
    {" ", "\\"}
};

extern const string detectorStates[2] = {"-", "0"};

extern const string printerStates[2] = {
    "  ||  ",
    "/====\\"
};

extern const string boxTypes[4][2] = {
    {
        "",
        "____"
    },
    {
        "*i^o",
        "____"
    },
    {
        " __ ",
        "[##]"
    },
    {
        " __ ",
        "[{}]"
    },
};

extern const string beltStates[3] = {
    "-==-==-==-==-=====-=====-==-==-==-=====-=====-==-==-==-",
    "==-==-=====-==-==-==-==-==-=====-=====-==-==-==-=====-=",
    "=-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-==-=="
};
```

Листинг А.6 — Файл src/models.h

```
#ifndef models
#define models

#include <vector>
#include <string>

using namespace std;

struct ConveyorStatuses {
    int printer;
    int detailType;
    string detailData;
    int detailXCords;
    bool firstDetector;
    bool secondDetector;
    bool topContainer;
    bool bottomContainer;
    int topContainerValue;
    int bottomContainerValue;
    int errorIndex;
};

struct Language {
    string exit;

    string mainMenuTitle;
    vector<string> mainMenuEntries;

    string systemMenuTitle;
    string systemMenuStopButton;
    string systemMenuStartButton;
    string systemMenuConveyorStopButton;
    string systemMenuConveyorStartButton;
    vector<string> systemErrors;
    vector<string> systemMenuTabs;

    string settingsMenuTitle;
    string settingsMenuLanguage;
    string settingsMenuConatinerValue;
    string settingsMenuFirstConatiner;
    string settingsMenuSecondConatiner;
    string settingsMenuThirdConatiner;
    vector<string> settingsMenuLanguageEntries;

    string aboutMenuTitle;
    string aboutMenuCodename;
    string aboutMenuCreator;
};

struct Settings {
    int containerMaxValue;
    int firstContainerValue;
    int secondContainerValue;
    int thirdContainerValue;
    Language language;
};

extern const vector<Language> languages;

extern const string conveyorTemplate[13];
extern const string containerStates[2][2];
extern const string detectorStates[2];
```

```
extern const string printerStates[2];
extern const string boxTypes[4][2];
extern const string beltStates[3];

#endif
```

Листинг А.7 — Файл src/system.cpp

```
#include <ctime>
#include <thread>
#include <vector>
#include <string>

#include "models.h"
#include "system.h"

#define DETAIL_SIZE 4
#define PRINTER_CORDS 25
#define MAX_DETAIL_CORDS 47
#define FIRST_DETECTOR_CORDS 13
#define SECOND_DETECTOR_CORDS 42
#define STOREHOUSE_MAX_VALUE 999

using namespace std;

// TickGenerator class functions
void TickGenerator::run() {
    while (status) {
        this_thread::sleep_for(0.3s);
        currentTick++;
    }
}

TickGenerator::TickGenerator() : currentTick(0), status(true) {
    mainThread = thread(&TickGenerator::run, this);
}

TickGenerator::~TickGenerator() {
    if (!status) return;

    status = false;
    mainThread.join();
}

void TickGenerator::delay(int ticks) {
    int endTick = currentTick + ticks;
    while (endTick > currentTick) {}
}

// Detail class functions
Detail::Detail(int _typeIndex) : typeIndex(_typeIndex) {}

// Container class functions
DetailContainer::DetailContainer(int _maxNumberOfDetails) :
    maxNumberOfDetails(_maxNumberOfDetails), topDriverStatus(false),
    bottomDriverStatus(false) {}

DetailContainer::DetailContainer(int _maxNumberOfDetails, int numberOfDetails,
int detailTypeIndex) :
    maxNumberOfDetails(_maxNumberOfDetails), topDriverStatus(true),
bottomDriverStatus(false) {
    fillWithDetails(numberOfDetails, detailTypeIndex);
    topDriverStatus = false;
}
```

```
DetailContainer::~DetailContainer() {
    for (int index = 0; index < details.size(); index++)
        delete details[index];
    details.clear();
}

Detail *DetailContainer::getDatail() {
    if (details.empty())
        throw underflow_error("The container ran out of details.");

    if (!bottomDriverStatus)
        throw logic_error("The container lid is closed.");

    Detail *tmp = details.back();
    details.pop_back();

    return tmp;
}

void DetailContainer::addDetail(Detail *detail) {
    if (details.size() + 1 > maxNumberOfDetails)
        throw out_of_range("The maximum level of details in the container has
been exceeded.");

    if (!topDriverStatus)
        throw logic_error("The container lid is closed.");

    details.push_back(detail);
}

void DetailContainer::fillWithDetails(int numberOfDetails, int detailTypeIndex)
{
    if (numberOfDetails + details.size() > maxNumberOfDetails)
        throw out_of_range("The maximum level of details in the container has
been exceeded.");

    for (int index = 0; index < numberOfDetails; index++)
        addDetail(new Detail(detailTypeIndex));
}

int DetailContainer::getNumberOfDetails() { return details.size(); }

int DetailContainer::getMaxNumberOfDetails() { return maxNumberOfDetails; }

// Belt class functions
void Belt::run() {
    if (!detail) {
        status = false;
        throw logic_error("There are no details on the conveyor belt.");
    }

    while (status && detailXCords < MAX_DETAIL_CORDS) {
        tickGenerator->delay(1);
        detailXCords++;
    }
}

Belt::Belt(TickGenerator *_tickGenerator) :
    status(false), detailXCords(0), detail(nullptr),
tickGenerator(_tickGenerator) {}

Belt::~Belt() { stop(); }

void Belt::stop() {
```

```
    if (!status) return;

    status = false;
    mainThread.join();
}

void Belt::start() {
    if (status) return;

    status = true;
    mainThread = thread(&Belt::run, this);
}

int Belt::getDetailCords() {
    if (!detail)
        throw underflow_error("There is no detail on the belt.");

    return detailXCords;
}

Detail *Belt::getDetail() {
    if (!detail)
        throw underflow_error("There is no detail on the belt.");

    return detail;
}

Detail *Belt::pickUpDetail() {
    if (!detail)
        throw underflow_error("There is no detail on the belt.");

    if (detailXCords < MAX_DETAIL_CORDS)
        throw logic_error("The detal didn't go all the way through.");

    Detail *tmpDetail = detail;
    detail = nullptr;
    stop();

    return tmpDetail;
}

void Belt::putDetail(Detail *newDetail) {
    if (!newDetail)
        throw overflow_error("There's already a detail on the belt.");

    detail = newDetail;
    detailXCords = 0;
}

// Printer class functions
Printer::Printer(Belt *_belt, TickGenerator *_tickGenerator, void
(*_printFunction)(Detail *)) :
    printerStatus(0), status(true), belt(_belt), tickGenerator(_tickGenerator),
    printFunction(_printFunction) {};

void Printer::print() {
    for (; (printerStatus < 2) && status; printerStatus++)
        tickGenerator->delay(1);

    if (!status) return;
    printFunction(belt->getDetail());

    for (; (printerStatus > 0) && status; printerStatus--)
        tickGenerator->delay(1);
```

```
}

void Printer::stop() { status = false; }

void Printer::start() {
    status = true;
    for (; (printerStatus > 0) && status; printerStatus--)
        tickGenerator->delay(1);
}

int Printer::getPrinterStatus() { return printerStatus; }

// Detector class functions
Detector::Detector(Belt *_belt, int _detectableCords) :
    belt(_belt), detectableCords(_detectableCords) {};

bool Detector::getStatus() {
    try {
        return belt->getDetailCords() >= detectableCords &&
            belt->getDetailCords() < detectableCords + DETAIL_SIZE;
    }
    catch (const underflow_error& error) { return false; }
}

// Conveyor class functions
void Conveyor::run() {
    /*
      \    /\
       )  ( ')
      (  /  )
       \(__)|

    I'm a little kitten who's tired of writing code.
    Please do not pick on the crutches written below, meow :3
    */

    try {
        int detailCords = belt->getDetailCords();
        if ((detailCords > 0 && detailCords < 25) ||
            (detailCords > 26 && detailCords < MAX_DETAIL_CORDS)) belt->start();
    }
    catch (const underflow_error& error) {}

    while (status) {
        int detailCords;
        try { detailCords = belt->getDetailCords(); }
        catch (const underflow_error& error) { detailCords = -1; }

        if (detailCords == -1 && !topContainer->bottomDriverStatus) {
            if (topContainer->getNumberOfDetails() <= 0) {
                errorIndex = 1; // The container ran out of details.
                break;
            }

            topContainer->bottomDriverStatus = true;
        }
        else if (detailCords == -1 && topContainer->bottomDriverStatus)
            belt->putDetail(topContainer->getDatail());
        else if (detailCords == 0 && topContainer->bottomDriverStatus)
            topContainer->bottomDriverStatus = false;
        else if (detailCords == 0 && !topContainer->bottomDriverStatus)
            belt->start();
        else if (detailCords == PRINTER_CORDS) {
            belt->stop();
```

```
            tickGenerator->delay(1);
            if (!status) break;
            printer->print();
            tickGenerator->delay(1);
            if (!status) break;
            belt->start();
        }
        else if (secondDetector->getStatus())
            bottomContainer->topDriverStatus = true;
        else if (detailCords == MAX_DETAIL_CORDS) {
            if (bottomContainer->getMaxNumberOfDetails() < bottomContainer-
>getNumberOfDetails() + 1) {
                errorIndex = 2; // The maximum level of details in the container
has been exceeded.
                break;
            }

            bottomContainer->addDetail(belt->pickUpDetail());
            tickGenerator->delay(1);
            if (!status) break;
            bottomContainer->topDriverStatus = false;

            if (bottomContainer->getMaxNumberOfDetails() <= bottomContainer-
>getNumberOfDetails()) {
                errorIndex = 2; // The maximum level of details in the container
has been exceeded.
                break;
            }
        }
        tickGenerator->delay(1);
    }

    status = false;
    belt->stop();
    printer->stop();
}

Conveyor::Conveyor(DetailContainer *_topContainer, DetailContainer
*_bottomContainer,
    TickGenerator *_tickGenerator, void (*printFunction)(Detail *)) :
topContainer(_topContainer),
    bottomContainer(_bottomContainer), tickGenerator(_tickGenerator),
status(false), errorIndex(0) {
    belt = new Belt(tickGenerator);
    printer = new Printer(belt, tickGenerator, printFunction);
    firstDetector = new Detector(belt, FIRST_DETECTOR_CORDS);
    secondDetector = new Detector(belt, SECOND_DETECTOR_CORDS);
}

Conveyor::~Conveyor() {
    belt->stop();
    printer->stop();
    try { mainThread.join(); }
    catch (const exception& e){}

    delete belt;
    delete printer;
    delete firstDetector;
    delete secondDetector;
}

void Conveyor::stop() {
    if (!status) return;
```

```
    status = false;
    belt->stop();
    printer->stop();
    mainThread.join();
}

void Conveyor::start() {
    if (status) return;

    try { mainThread.join(); }
    catch (const exception& e){}

    status = true;
    errorIndex = 0;
    printer->start();
    mainThread = thread(&Conveyor::run, this);
}

ConveyorStatuses Conveyor::getStatuses() {
    ConveyorStatuses statuses;

    try {
        Detail *tmpDetail = belt->getDetail();
        statuses.detailXCords = belt->getDetailCords();
        statuses.detailType = tmpDetail->typeIndex;
        statuses.detailData = tmpDetail->data;
    }
    catch (const underflow_error& error) {
        statuses.detailXCords = -1;
        statuses.detailType = -1;
        statuses.detailData = "";
    }

    statuses.errorIndex = errorIndex;
    statuses.printer = printer->getPrinterStatus();
    statuses.firstDetector = firstDetector->getStatus();
    statuses.secondDetector = secondDetector->getStatus();
    statuses.topContainer = topContainer->bottomDriverStatus;
    statuses.bottomContainer = bottomContainer->topDriverStatus;
    statuses.topContainerValue = topContainer->getNumberOfDetails();
    statuses.bottomContainerValue = bottomContainer->getNumberOfDetails();

    return statuses;
}

// System class functions
void System::stop() {
    firstConveyor->stop();
    secondConveyor->stop();
    thirdConveyor->stop();
}

void System::start() {
    firstConveyor->start();
    secondConveyor->start();
    thirdConveyor->start();
}

System::System(const Settings settings) {
    tickGenerator = new TickGenerator();

    firstContainer = new DetailContainer(settings.containerMaxValue,
settings.firstContainerValue, 0);
```

```
    secondContainer = new DetailContainer(settings.containerMaxValue,
settings.secondContainerValue, 1);
    thirdContainer = new DetailContainer(settings.containerMaxValue,
settings.thirdContainerValue, 2);
    storehouseContainer = new DetailContainer(STOREHOUSE_MAX_VALUE);

    firstConveyor = new Conveyor(firstContainer, secondContainer, tickGenerator,
        [](Detail* detail){ detail->typeIndex = 1; });
    secondConveyor = new Conveyor(secondContainer, thirdContainer,
tickGenerator,
        [](Detail* detail){ detail->typeIndex = 2; });
    thirdConveyor = new Conveyor(thirdContainer, storehouseContainer,
tickGenerator,
        [](Detail* detail){
            srand(time(0));
            detail->typeIndex = 3;
            detail->data = char(65 + rand() % 26) + to_string(rand() % 10);
        });
}

System::~System() {
    stop();

    delete tickGenerator;

    delete firstContainer;
    delete secondContainer;
    delete thirdContainer;
    delete storehouseContainer;

    delete firstConveyor;
    delete secondConveyor;
    delete thirdConveyor;
}
```

Листинг А.8 — Файл src/system.h

```
#ifndef system
#define system

#include <thread>
#include <vector>
#include <string>

#include "models.h"

class TickGenerator {
private:
    bool status;
    int currentTick;
    thread mainThread;

    void run();

public:
    TickGenerator();
    ~TickGenerator();

    void delay(int ticks);
};

class Detail {
public:
    string data;
    int typeIndex;
```

```
    Detail(int _typeIndex);
};

class DetailContainer {
private:
    int maxNumberOfDetails;
    vector<Detail *> details;

public:
    bool topDriverStatus;
    bool bottomDriverStatus;

    DetailContainer(int _maxNumberOfDetails);
    DetailContainer(int _maxNumberOfDetails, int numberOfDetails, int
detailTypeIndex);
    ~DetailContainer();

    int getNumberOfDetails();
    int getMaxNumberOfDetails();
    Detail *getDatail();

    void addDetail(Detail *detail);
    void fillWithDetails(int numberOfDetails, int detailTypeIndex);
};

class Belt {
private:
    bool status;
    int detailXCords;
    thread mainThread;

    Detail *detail;
    TickGenerator *tickGenerator;

    void run();

public:
    Belt(TickGenerator *_tickGenerator);
    ~Belt();

    void stop();
    void start();

    int getDetailCords();
    Detail *getDetail();
    Detail *pickUpDetail();

    void putDetail(Detail *detail);
};

class Printer {
private:
    int status;
    int printerStatus;

    Belt *belt;
    TickGenerator *tickGenerator;
    void (*printFunction)(Detail *);

public:
    Printer(Belt *_belt, TickGenerator *_tickGenerator, void
(*_printFunction)(Detail *));
```

```
    void stop();
    void start();

    void print();

    int getPrinterStatus();
};

class Detector {
private:
    int detectableCords;

    Belt *belt;

public:
    Detector(Belt *_belt, int _detectableCords);

    bool getStatus();
};

class Conveyor {
private:
    thread mainThread;
    int errorIndex;

    Belt *belt;
    Printer *printer;
    Detector *firstDetector;
    Detector *secondDetector;
    DetailContainer *topContainer;
    DetailContainer *bottomContainer;
    TickGenerator *tickGenerator;

    void run();

public:
    bool status;

    Conveyor(DetailContainer *_topContainer, DetailContainer *_bottomContainer,
        TickGenerator *_tickGenerator, void (*printFunction)(Detail *));
    ~Conveyor();

    void stop();
    void start();

    ConveyorStatuses getStatuses();
};

class System {
private:
    TickGenerator *tickGenerator;

    DetailContainer *firstContainer;
    DetailContainer *secondContainer;
    DetailContainer *thirdContainer;
    DetailContainer *storehouseContainer;

public:
    Conveyor *firstConveyor;
    Conveyor *secondConveyor;
    Conveyor *thirdConveyor;

    System(const Settings settings);
    ~System();
```

```
    void stop();
    void start();
};

#endif
```

Листинг А.1 — Файл CMakeList.txt

```
cmake_minimum_required (VERSION 3.11)

# --- Dependencies -------------------------------------------------------------
include(FetchContent)

FetchContent_Declare(ftxui
  GIT_REPOSITORY https://github.com/ArthurSonzogni/ftxui
  GIT_TAG v5.0.0
  GIT_PROGRESS TRUE
  GIT_SHALLOW FALSE
)
FetchContent_MakeAvailable(ftxui)

FetchContent_Declare(fmt
  GIT_REPOSITORY https://github.com/fmtlib/fmt
  GIT_TAG master
)
FetchContent_MakeAvailable(fmt)
# ------------------------------------------------------------------------------

project(factory-emulator
  LANGUAGES CXX
  VERSION 1.0.0
)

add_executable(factory-emulator
  src/main.cpp
  src/models.cpp
  src/models.h
  src/system.cpp
  src/system.h
  src/interface.cpp
  src/interface.h
  src/environment.h
)

target_include_directories(factory-emulator PRIVATE src)

target_compile_features(factory-emulator PRIVATE cxx_std_20)

target_link_libraries(factory-emulator
  PRIVATE ftxui::screen
  PRIVATE ftxui::dom
  PRIVATE ftxui::component
  PRIVATE fmt::fmt
)
```