**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования «Воронежский государственный технический университет»**

**(ФГБОУ ВО «ВГТУ», ВГТУ)**

**Факультет** экономики, менеджмента и информационных технологий

**Кафедра** систем управления и информационных технологий в строительстве

**Курсовой проект**

по дисциплине: «Программирование мобильных приложений»

на тему: «Разработка программного обеспечения демонстрирующие работу завода по изготовлению деталей»

Выполнил студент: Чайкин Д. А.

Группа: бИС - 181

Руководитель: доцент, к.т.н. Черменев Д.А.

Работа защищена « » 2022 г.

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Воронеж

2022

**СОДЕРЖАНИЕ**

[1. Анализ предметной области 4](#_Toc92733462)

[2. Анализ средств разработки 6](#_Toc92733463)

[2.1 Многопоточность. Синхронизация потоков 6](#_Toc92733464)

[2.2 Рефлексия. Делегаты и события 7](#_Toc92733465)

[3. Разработка программного обеспечения 9](#_Toc92733466)

[3.1 Выбор средств реализации 9](#_Toc92733467)

[3.2 Пример работы программного обеспечения 10](#_Toc92733468)

[ЗАКЛЮЧЕНИЕ 12](#_Toc92733469)

[Приложение 14](#_Toc92733471)

ВВЕДЕНИЕ

В современном мире просто невозможно представить жизнь без информационных технологий. Информационные технологии представляют весь накопленный опыт человечества в форматизированном виде, пригодном для прикладного использования. Сегодня информационные технологии, которые улучшают условия жизни людей, стремительно развиваются. Значительно повышается уровень знаний, в связи с этим становится необходимым эффективно организовывать, сохранять и управлять доступом к ним. Вопрос о потребности в автоматизации является актуальным в наше время, в решении которого, несомненно, помогает программирование. Одной из основных задач можно рассматривать проблему ведения отчетности, а также оперативную корректировку данных при возникновении необходимости в этом. Для работы с данными используются системы управления базами данных.

Целью данного курсового проекта является разработка программного обеспечения демонстрирующего работу завода по изготовлению деталей.

Станок по производству деталей – смоделировать работу станка. Реализовать классы – Станок, Фрезеровщик, интерфейс – погрузчик. События – работать с новой деталью – Фрезеровщик берет новую деталь и работает с ней, Деталь готова – погрузчик забирает деталь.

Программа разработана в среде MS Visual Studio 2019.

# Анализ предметной области

Темой данного курсового проекта является разработка программного обеспечения демонстрирующие работу завода по изготовлению деталей.

Завод – промышленное предприятие, обычно с механизированными производственными процессами.

Фрезерный станок –группа [металлорежущих](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%B0%D0%BB%D0%BB%D0%BE%D1%80%D0%B5%D0%B6%D1%83%D1%89%D0%B8%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA) и [деревообрабатывающих](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%B0%D1%82%D1%8B%D0%B2%D0%B0%D1%8E%D1%89%D0%B8%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%BA) станков в классификации по виду обработки. Фрезерные станки предназначены для обработки с помощью [фрезы](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B7%D0%B0) плоских и фасонных поверхностей, [зубчатых колёс](https://ru.wikipedia.org/wiki/%D0%97%D1%83%D0%B1%D1%87%D0%B0%D1%82%D1%8B%D0%B5_%D0%BA%D0%BE%D0%BB%D0%B5%D1%81%D0%B0) и т. п., металлических и других заготовок. При этом фреза, закрепленная в [шпинделе](https://ru.wikipedia.org/wiki/%D0%A8%D0%BF%D0%B8%D0%BD%D0%B4%D0%B5%D0%BB%D1%8C) фрезерного станка, совершает [вращательное (главное) движение](https://ru.wikipedia.org/wiki/%D0%92%D1%80%D0%B0%D1%89%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B2%D0%B8%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), а [заготовка](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%BA%D0%B0_(%D0%BC%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%B0%D0%BB)), закреплённая на столе, совершает [движение подачи](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%BE%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D1%83%D1%8E%D1%89%D0%B8%D0%B5_%D0%B4%D0%B2%D0%B8%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F), прямолинейное или криволинейное (иногда осуществляется одновременно вращающимся инструментом). Управление может быть ручным, автоматизированным или осуществляться с помощью системы [ЧПУ](https://ru.wikipedia.org/wiki/%D0%A7%D0%9F%D0%A3).

Погрузчик — самоходная или ручная (мускульная) [машина](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0) для поднятия, [транспортировки](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BF%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0) и укладки различных [грузов](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D1%83%D0%B7).

Склад – [территория](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%80%D1%80%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F), [помещение](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BC%D0%B5%D1%89%D0%B5%D0%BD%D0%B8%D0%B5) (также их комплекс), предназначенное для хранения [материальных ценностей](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%86%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8) и оказания складских услуг. Склады используются производителями, импортёрами, экспортёрами, оптовыми торговцами, транспортными предприятиями, таможней и т. д.

Постановка задачи:

В зависимости от задачи необходимо смоделировать ситуацию/процесс. В каждой модели есть набор возможных ситуаций. Для некоторых событий необходимо определить вероятность возникновения данного события. Интерфейс необходимо реализовать, используя 3 и более классов с использованием синхронизации потоков. На форме должно быть не менее 4 моделей. Ограничения накладываются на классы, которые реализуют интерфейсы. Для 4 моделей должно быть 2 объекта данных классов в сумме. При возникновении какого-то события 1 из объектов «лочится» и не доступен для использования в других моделях.

Для решения задач необходимо использовать:

1. Делегаты/события.

2. Многопоточность

3. Где необходимо рефлексию

4. Синхронизацию потоков

На форме должно быть динамическое изменение моделей – все должно двигаться. Иметь возможность добавлять несколько моделей на форму

# Анализ средств разработки

## 2.1 Многопоточность. Синхронизация потоков

Одним из ключевых аспектов в современном программировании является многопоточность. Ключевым понятием при работе с многоопоточностью является поток. Поток предствляет некоторую часть кода программы. При выполнении программы каждому потоку выделяется определенный квант времени. И при помощи многопоточности мы можем выделить в приложении несколько потоков, которые будут выполнять различные задачи одновременно. Если у нас, допустим, графическое приложение, которое посылает запрос к какому-нибудь серверу или считывает и обрабатывает огромный файл, то без многопоточности у нас бы блокировался графический интерфейс на время выполнения задачи. А благодаря потокам мы можем выделить отправку запроса или любую другую задачу, которая может долго обрабатываться, в отдельный поток. Поэтому, к примеру, клиент-серверные приложения (и не только они) практически не мыслимы без многопоточности.

Основной функционал для использования потоков в приложении сосредоточен в пространстве имен System.Threading. В нем определен класс, представляющий отдельный поток - класс Thread.Процесс — это исполнение программы. Операционная система использует процессы для разделения исполняемых приложений.

Поток — это основная единица, которой операционная система выделяет время процессора. Каждый поток имеет [приоритет планирования](https://docs.microsoft.com/ru-ru/dotnet/standard/threading/scheduling-threads) и набор структур, в которых система сохраняет контекст потока, когда выполнение потока приостановлено. Контекст потока содержит все сведения, позволяющие потоку безболезненно возобновить выполнение, в том числе набор регистров процессора и стек потока. Несколько потоков могут выполняться в контексте процесса. Все потоки процесса используют общий диапазон виртуальных адресов. Поток может исполнять любую часть программного кода, включая части, выполняемые в данный момент другим потоком.

Платформа .NET предоставляет множество готовых примитивов синхронизации и потокобезопасных коллекций. Если при разработке приложения нужно реализовать, например, потокобезопасный кэш или очередь запросов — обычно используются эти готовые решения, иногда сразу несколько. В отдельных случаях это приводит к проблемам с производительностью: долгим ожиданием на блокировках, избыточному потреблению памяти и долгим сборкам мусора.

Синхронизация потоков это процесс, который позволяет выполнять все параллельные потоки в программе синхронно. Синхронизация позволяет избежать ошибок согласованности памяти, вызванные из-за непоследовательного доступа к общей памяти.

При выполнении нескольких потоков они будут неизменно взаимодействовать друг с другом, чтобы синхронизировать свое выполнение. Существует несколько методов синхронизации потоков:

* Блоки взаимного исключения;
* Переменные состояния;
* Семафоры.

## 2.2 Рефлексия. Делегаты и события

В [информатике](https://wikipedia.tel/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) отражение или рефлексия ([холоним](https://wikipedia.tel/%D0%9C%D0%B5%D1%80%D0%BE%D0%BD%D0%B8%D0%BC_%D0%B8_%D1%85%D0%BE%D0%BB%D0%BE%D0%BD%D0%B8%D0%BC) [интроспекции](https://wikipedia.tel/%D0%98%D0%BD%D1%82%D1%80%D0%BE%D1%81%D0%BF%D0%B5%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), [англ.](https://wikipedia.tel/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) reflection) означает процесс, во время которого [программа](https://wikipedia.tel/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0) может отслеживать и модифицировать собственную структуру и поведение во время выполнения. [Парадигма программирования](https://wikipedia.tel/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), положенная в основу отражения, называется рефлексивным программированием.

Во время выполнения программных инструкций (кода) компьютеры обрабатывают данные, что приводит к их изменению, при этом компьютеры не изменяют код. Однако, в большинстве современных компьютерных архитектур код хранится как данные, и в некоторых языках программирования реализована возможность обрабатывать собственный код как данные, что приводит к изменению уже самого кода во время его выполнения.

Событие представляет собой автоматическое уведомление о том, что произошло некоторое действие. События действуют по следующему принципу: объект, проявляющий интерес к событию, регистрирует обработчик этого события. Когда же событие происходит, вызываются все зарегистрированные обработчики этого события. Обработчики событий обычно представлены делегатами.

Делегаты представляют такие объекты, которые указывают на методы. То есть делегаты - это указатели на методы и с помощью делегатов мы можем вызвать данные методы.

# Разработка программного обеспечения

## Выбор средств реализации

Программный продукт разрабатывается на языке программирования C# с использованием интерфейс программирования приложений Windows Forms.

Windows Forms — [интерфейс программирования приложений](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) (API), отвечающий за [графический интерфейс пользователя](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F) и являющийся частью [Microsoft](https://ru.wikipedia.org/wiki/Microsoft) [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework). Данный интерфейс упрощает доступ к элементам интерфейса [Microsoft Windows](https://ru.wikipedia.org/wiki/Microsoft_Windows) за счет создания обёртки для существующего [Win32 API](https://ru.wikipedia.org/wiki/Windows_API) в [управляемом коде](https://ru.wikipedia.org/wiki/%D0%A3%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%8F%D0%B5%D0%BC%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4). Причём управляемый код — классы, реализующие [API](https://ru.wikipedia.org/wiki/API) для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

С одной стороны, Windows Forms рассматривается как замена более старой и сложной библиотеке [MFC](https://ru.wikipedia.org/wiki/Microsoft_Foundation_Classes), изначально написанной на языке [C++](https://ru.wikipedia.org/wiki/C%2B%2B). С другой стороны, WF не предлагает парадигмы, сравнимой с [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller). Для исправления этой ситуации и реализации данной функциональности в WF существуют сторонние библиотеки. Одной из наиболее используемых подобных библиотек является [User Interface Process Application Block](http://www.microsoft.com/downloads/details.aspx?familyid=98C6CC9D-88E1-4490-8BD6-78092A0F084E&displaylang=en), выпущенная специальной группой Microsoft, занимающейся примерами и рекомендациями, для бесплатного скачивания. Эта библиотека также содержит исходный код и обучающие примеры для ускорения обучения.

Приложение Windows Forms представляет собой [событийно-ориентированное приложение](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), поддерживаемое Microsoft [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework). В отличие от [пакетных программ](https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D0%BA%D0%B5%D1%82%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0), большая часть времени тратится на ожидание от пользователя каких-либо действий, как, например, ввод текста в [текстовое поле](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D0%BE%D0%B5_%D0%BF%D0%BE%D0%BB%D0%B5) или клика мышкой по [кнопке](https://ru.wikipedia.org/wiki/%D0%9A%D0%BD%D0%BE%D0%BF%D0%BA%D0%B0_(%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)).

## Пример работы программного обеспечения

Рассмотрим работу созданного программного продукта. Начальная страница приложения представлена на рисунке 1.

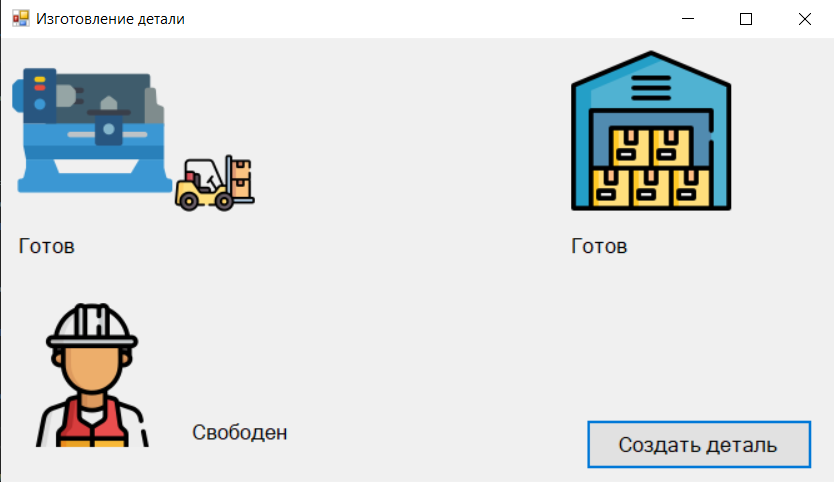


Рисунок 1 – Начальная страница

В данном окне расположены картинки с изображениями рабочего, фрезерного станка, склада и погрузчика. Строки состояния и кнопка для начала создания детали.

После нажатия кнопки «Создать деталь» статус фрезеровщика изменится на «Деталь принял в работу», а статус станка поменяется на «Станок изготавливает деталь». Смотреть рисунок 2.

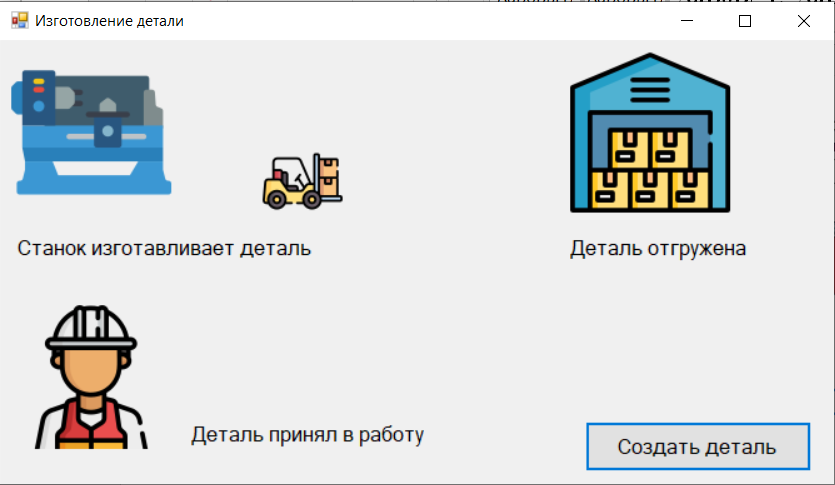


Рисунок 2 – Состояние программного обеспечения после начала работы

Когда станок завершит работу его статус поменяется на «Деталь готова», деталь будет передана погрузчику, который отвезет деталь на склад и статус склада изменится на «Деталь отгружена». Смотреть рисунок 3.

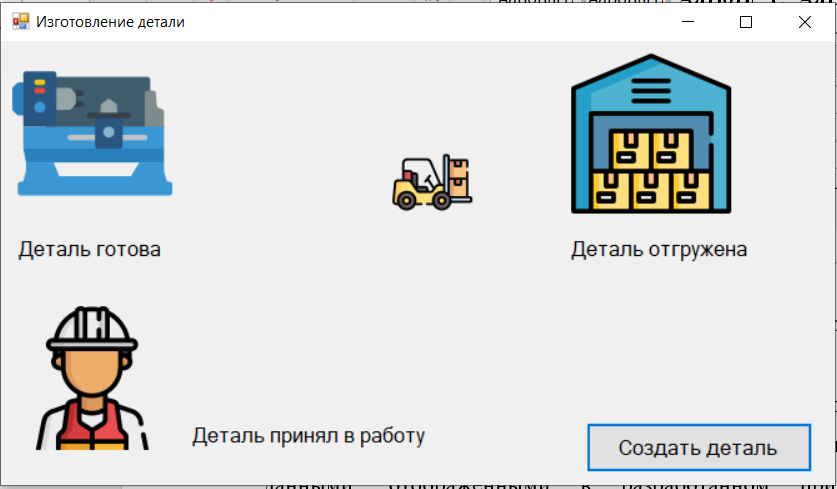


Рисунок 3 – Работа погрузчика

После отгрузки детали статус фрезеровщика изменится на «Свободен»

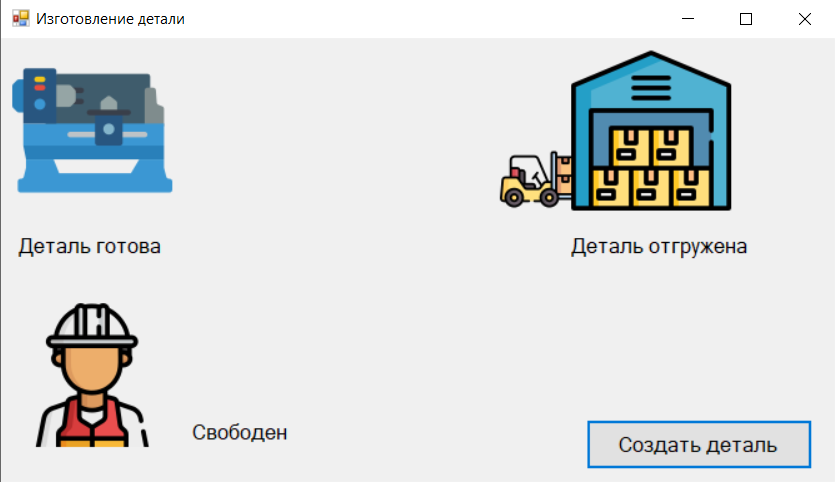


Рисунок 4 – Завершенный процесс изготовления детали

Таким образом, были рассмотрены основные возможности приложения был разобран функционал, представленный на каждой странице. Производимые команды и действия с данными были проиллюстрированы на примерах. Были описаны основные операции, которые можно производить с данными, отображенными в разработанном приложении. Листинг программного обеспечения приведен в приложении.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были реализованы основные задачи курсового проекта Станок по производству деталей – смоделирована работа станка. Реализованы классы – Станок, Фрезеровщик, интерфейс – погрузчик. События – работать с новой деталью – Фрезеровщик берет новую деталь и работает с ней, Деталь готова – погрузчик забирает деталь. Задача была реализована на языке программирования высокого уровня C#. Программа разработана в среде Microsoft Visual Studio 2019.

Были приобретены навыки самостоятельной разработки программного обеспечения, которые могут быть использованы в разных предметных областях.

# Список используемых источников.

1. EDM (модель данных с использованием сущностей). [Электронный ресурс]/ Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/entity-data-model>
2. Потоки. [Программирование](https://habr.com/ru/hub/programming/) [C++](https://habr.com/ru/hub/cpp/) [Разработка игр](https://habr.com/ru/hub/gamedev/) [Параллельное программирование/ ). [Электронный ресурс]/](https://habr.com/ru/hub/parallel_programming/)  Режим доступа: <https://habr.com/ru/post/279653/>
3. Чтение и запись текстовых файлов. StreamReader и StreamWriter [Электронный ресурс]/ Режим доступа: https://metanit.com/sharp/tutorial/5.5.php
4. XAML. Введение в язык XAML. [Электронный ресурс]/ Режим доступа: <https://metanit.com/sharp/wpf/2.php>
5. Орлов С.А. Технологии разработки программного обеспечения: учебное пособие/ Орлов С.А. – М.: Диалог – Наука, 2003. –477 с.

# Приложение

Листинг программного кода

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Factory

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

using System;

using System.Windows.Forms;

using System.Threading;

using System.Drawing;

namespace Factory

{

public partial class Form1 : Form

{

private Machine \_auto;

private Miller[] \_millers;

private Loader \_loader;

private ManualResetEvent pogrNew1;

private ManualResetEvent pogrReady;

private ManualResetEvent evToLoaderpogr;

private ManualResetEvent MachineWork;

private static Semaphore \_pool;

public Form1()

{

InitializeComponent();

pictureBox1.Image = Image.FromFile("C:\\Users\\admin\\Desktop\\WindowsFormsApp2\\Images\\Image1.png");

pictureBox2.Image = Image.FromFile("C:\\Users\\admin\\Desktop\\WindowsFormsApp2\\Images\\Image4.png");

pictureBox3.Image = Image.FromFile("C:\\Users\\admin\\Desktop\\WindowsFormsApp2\\Images\\Image6.png");

item.Image = Image.FromFile("C:\\Users\\admin\\Desktop\\WindowsFormsApp2\\Images\\Image5.png");

CheckForIllegalCrossThreadCalls = false;

pogrNew1 = new ManualResetEvent(false);

pogrReady = new ManualResetEvent(false);

MachineWork = new ManualResetEvent(false);

evToLoaderpogr = new ManualResetEvent(false);

\_pool = new Semaphore(1, 1);

\_millers = new Miller[1];

\_millers[0] = new Miller(ref people1, \_pool, pogrNew1, pogrReady, MachineWork, evToLoaderpogr);

\_auto = new Machine(ref statusMachine, ref item, MachineWork, pogrReady);

\_loader = new Loader(ref loader, evToLoaderpogr);

}

private void newItem(object sender, EventArgs e)

{

Button PushButtom = sender as Button;

switch (PushButtom.Text)

{

case "Создать деталь":

pogrNew1.Set();

break;

default:

return;

}

}

}

interface ILoader

{

void receivedItem();

void loadedItem();

}

class Miller

{

public Thread Thrd;

ManualResetEvent ItemNew;

ManualResetEvent ItemReady;

ManualResetEvent MachineWork;

ManualResetEvent ItemToLoader;

private Semaphore MachineIsBusy;

private Label text;

public Miller( ref Label field,

Semaphore MachineIsBusy,

ManualResetEvent ItemNew,

ManualResetEvent ItemReady,

ManualResetEvent MachineWork,

ManualResetEvent ItemToLoader )

{

text = field;

this.MachineIsBusy = MachineIsBusy;

this.ItemNew = ItemNew;

this.ItemReady = ItemReady;

this.MachineWork = MachineWork;

this.ItemToLoader = ItemToLoader;

Thrd = new Thread(Run);

Thrd.Start();

}

void Run()

{

while (true)

{

text.Text = "Свободен";

ItemNew.WaitOne();

MachineIsBusy.WaitOne();

text.Text = "Деталь принял в работу";

MachineWork.Set();

ItemReady.WaitOne();

text.Text = "Деталь готова";

ItemNew.Reset();

ItemReady.Reset();

MachineIsBusy.Release();

ItemToLoader.Set();

}

}

}

class Loader : ILoader

{

public Thread Thrd;

public ManualResetEvent evToLoader;

public Label text;

public Loader(ref Label field, ManualResetEvent evToLoader)

{

Thrd = new Thread(this.Run);

this.evToLoader = evToLoader;

text = field;

Thrd.Start();

}

public void Run()

{

while (true)

{

evToLoader.WaitOne();

receivedItem();

Thread.Sleep(1000);

loadedItem();

evToLoader.Reset();

}

}

public void receivedItem()

{

text.Text = "Погрузка детали";

}

public void loadedItem()

{

text.Text = "Деталь отгружена";

}

}

class Machine

{

public Thread Thrd;

public ManualResetEvent MachineIsBusy;

public ManualResetEvent ItemReady;

public Label text;

public PictureBox Thing;

public Machine(ref Label field, ref PictureBox Thing, ManualResetEvent MachineIsBusy, ManualResetEvent ItemReady)

{

Thrd = new Thread(Run);

this.MachineIsBusy = MachineIsBusy;

this.ItemReady = ItemReady;

this.Thing = Thing;

text = field;

Thrd.Start();

}

public void Run()

{

while (true)

{

MachineIsBusy.WaitOne();

int start = 210;

int end = 400;

int Y = 81;

int X = start;

int step = 1;

Thing.Location = new Point(210, 81);

text.Text = "Станок изготавливает деталь";

Thread.Sleep(5000);

text.Text = "Деталь готова";

while (X < end)

{

Thing.Location = new Point(X, Y);

X += step;

Thread.Sleep(5);

}

MachineIsBusy.Reset();

ItemReady.Set();

}

}

}

}

namespace Factory

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.bPeople1 = new System.Windows.Forms.Button();

this.statusMachine = new System.Windows.Forms.Label();

this.people1 = new System.Windows.Forms.Label();

this.loader = new System.Windows.Forms.Label();

this.pictureBox1 = new System.Windows.Forms.PictureBox();

this.item = new System.Windows.Forms.PictureBox();

this.pictureBox2 = new System.Windows.Forms.PictureBox();

this.pictureBox3 = new System.Windows.Forms.PictureBox();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.item)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox2)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox3)).BeginInit();

this.SuspendLayout();

//

// bPeople1

//

this.bPeople1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.bPeople1.Location = new System.Drawing.Point(624, 375);

this.bPeople1.Margin = new System.Windows.Forms.Padding(4);

this.bPeople1.Name = "bPeople1";

this.bPeople1.Size = new System.Drawing.Size(241, 49);

this.bPeople1.TabIndex = 4;

this.bPeople1.Text = "Создать деталь";

this.bPeople1.UseVisualStyleBackColor = true;

this.bPeople1.Click += new System.EventHandler(this.newItem);

//

// statusMachine

//

this.statusMachine.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.statusMachine.Location = new System.Drawing.Point(13, 192);

this.statusMachine.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.statusMachine.Name = "statusMachine";

this.statusMachine.Size = new System.Drawing.Size(348, 70);

this.statusMachine.TabIndex = 6;

this.statusMachine.Text = "Готов";

//

// people1

//

this.people1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.people1.Location = new System.Drawing.Point(199, 375);

this.people1.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.people1.Name = "people1";

this.people1.Size = new System.Drawing.Size(281, 48);

this.people1.TabIndex = 8;

this.people1.Text = "Ожидаю";

//

// loader

//

this.loader.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.loader.Location = new System.Drawing.Point(603, 192);

this.loader.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.loader.Name = "loader";

this.loader.Size = new System.Drawing.Size(433, 48);

this.loader.TabIndex = 16;

this.loader.Text = "Готов";

//

// pictureBox1

//

this.pictureBox1.Location = new System.Drawing.Point(12, 12);

this.pictureBox1.Name = "pictureBox1";

this.pictureBox1.Size = new System.Drawing.Size(128, 128);

this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.AutoSize;

this.pictureBox1.TabIndex = 18;

this.pictureBox1.TabStop = false;

//

// item

//

this.item.Enabled = false;

this.item.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.item.Location = new System.Drawing.Point(185, 104);

this.item.Margin = new System.Windows.Forms.Padding(4);

this.item.Name = "item";

this.item.Size = new System.Drawing.Size(64, 64);

this.item.SizeMode = System.Windows.Forms.PictureBoxSizeMode.AutoSize;

this.item.TabIndex = 19;

this.item.TabStop = false;

//

// pictureBox2

//

this.pictureBox2.Location = new System.Drawing.Point(608, 12);

this.pictureBox2.Name = "pictureBox2";

this.pictureBox2.Size = new System.Drawing.Size(128, 128);

this.pictureBox2.SizeMode = System.Windows.Forms.PictureBoxSizeMode.AutoSize;

this.pictureBox2.TabIndex = 20;

this.pictureBox2.TabStop = false;

//

// pictureBox3

//

this.pictureBox3.Location = new System.Drawing.Point(12, 261);

this.pictureBox3.Name = "pictureBox3";

this.pictureBox3.Size = new System.Drawing.Size(166, 142);

this.pictureBox3.TabIndex = 21;

this.pictureBox3.TabStop = false;

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(889, 437);

this.Controls.Add(this.pictureBox3);

this.Controls.Add(this.pictureBox2);

this.Controls.Add(this.item);

this.Controls.Add(this.pictureBox1);

this.Controls.Add(this.loader);

this.Controls.Add(this.people1);

this.Controls.Add(this.statusMachine);

this.Controls.Add(this.bPeople1);

this.Margin = new System.Windows.Forms.Padding(4);

this.Name = "Form1";

this.Text = "Изготовление детали";

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.item)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox2)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox3)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();

}

private System.Windows.Forms.Label loader;

private System.Windows.Forms.Button bPeople1;

private System.Windows.Forms.Label statusMachine;

private System.Windows.Forms.Label people1;

#endregion

private System.Windows.Forms.PictureBox pictureBox1;

private System.Windows.Forms.PictureBox item;

private System.Windows.Forms.PictureBox pictureBox2;

private System.Windows.Forms.PictureBox pictureBox3;

}

}