**Содержание**

Введение 6

1 Характеристика места практики 8

2 Сведения об имеющейся вычислительной технике и программном обеспечении 9

3 Сведения о прохождении практики в качестве дублера инженерно-технического работника 12

4 Индивидуальное задание 13

4.1 Постановка задачи 13

4.2 Проектирование структуры программы 14

4.2.1 Desktop версия приложения 14

4.2.2 Мобильная версия приложения 14

4.3 Функциональное взаимодействие модулей программы 15

4.3.1 Desktop версия приложения 15

4.3.2 Мобильная версия приложения 21

4.4 Разработка тестов 25

4.5 Работа с программой 30

4.5.1 Desktop версия приложения 30

4.5.2 Мобильная версия приложения 32

Заключение 35

Список использованных источников 36

**ВВЕДЕНИЕ**

Целью производственной практики по ПМ.01 Разработка модулей программного обеспечения для компьютерных систем является закрепление и совершенствование приобретенных в процессе обучения профессиональных умений обучающихся по специальности, развитие общих и профессиональных компетенций, адаптация к конкретным условиям деятельности предприятий и организаций, освоение современных производственных процессов, обучение трудовым приемам, операциям и способам выполнения трудовых процессов.

В процессе прохождения производственной практики необходимо будет решить следующие задачи:

* приобрести практический опыт разработки алгоритма решения поставленной задачи и реализации его средствами автоматизированного проектирования, использования инструментальных средств на этапе отладки программного продукта и проведения тестирования программного модуля по определенному сценарию;
* приобрести практический опыт анализа алгоритмов, в том числе с применением инструментальных средств, разработки кода программного продукта на основе готовой спецификации на уровне модуля, осуществления рефакторинга и оптимизации программного кода;
* приобрести практический опыт разработки мобильных приложений;
* сформировать профессиональные умения и навыки;
* приобрести практические навыки в будущей профессиональной деятельности при выполнении функций дублера инженерно-технических работников;
* выполнить индивидуальное задание.

В результате прохождения производственной практики необходимо будет подтвердить уровень овладения видом деятельности «Разработка модулей программного обеспечения для компьютерных систем» и следующими профессиональными компетенциями:

ПК 1.1 Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием.

ПК 1.2 Разрабатывать программные модули в соответствии с техническим заданием.

ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК 1.4 Выполнять тестирование программных модулей.

ПК 1.5 Осуществлять рефакторинг и оптимизацию программного кода.

ПК 1.6 Разрабатывать модули программного обеспечения для мобильных платформ.

Производственная практика проводится на базе ООО «Персональные системы».

**1 ХАРАКТЕРИСТИКА МЕСТА ПРАКТИКИ**

## Общество с ограниченной ответственностью « Персональные системы ».

## Основной вид деятельности организации: Разработка компьютерного программного обеспечения.

Юридический адрес ООО «Персональные системы»: 385000, Респ. Адыгея, г. Майкоп, ул. Заводская, д. 2.

Основными задачами ООО «Персональные системы» являются:

* разработка заказного программного обеспечения;
* системная интеграция ПО и оборудования (централизованные и распределенные системы, сопряжение ПО и оборудования);
* автоматизация финансово-хозяйственной деятельности предприятий и организаций (на платформе 1С Предприятие);
* системы спутникового позиционирования GPS, Глонасс (приложения, сервер мониторинга);
* web-программирование и разработка сайтов.

Официальный партнер 1С, Microsoft, GalileoSky.

Рабочее время организации:

Пн - пт: 08:00 – 17:00.

Обеденный перерыв: 12:00 – 13:00.

Директор ООО «Персональные системы» – Небольсин Эдуард Викторович, опытный разработчик в сфере веб-технологий.

Руководитель отдела, в котором проходила практика – Небольсин Эдуард Викторович.

**2 СВЕДЕНИЯ ОБ ИМЕЮЩЕЙСЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКЕ И ПРОГРАММНОМ ОБЕСПЕЧЕНИИ**

Практика проходила в IT-отделе организации ООО «Персональные системы».

IT-отдел имеет следующие характеристики:

* площадь (не менее 6 кв.м. на одно место) – 35 кв.м;
* электрощит – 1 шт.;
* электророзетки – 12 шт.;
* компьютерные столы– 4 шт.;
* расстановка рабочих мест с ПЭВМ – периметральная.

В помещении установлены 4 компьютеров со следующими характеристиками:

* материнская плата: ASUS;
* процессор: 3.21 ГГЦ;
* видео карта: GIGABYTE GeForce GT 1030;
* оперативная память (RAM): 16 ГБ;
* жесткий диск (HDD): 500 ГБ;

Компьютеры объединены в локальную сеть и имеют выход в Internet.

Программное обеспечение рабочего места:

* Microsoft Visio Professional 2019;
* Microsoft Office 2019;
* Git 2.32;
* .NET Framework Developer Pack 4.8;
* Microsoft Visual Studio 2022 Community, включая следующие компоненты:
  + .NET desktop development Workload;
  + Universal Windows Platform development Workload;
  + Data storage and processing Workload;
  + Entity Framework (EF); IIS, ASP.NET and web development, Mobile development with .NET;
* Android Studio 4.2.1, включая следующие компоненты:
  + Android SDK Tools;
  + Android SDK Platform-Tools;
  + Android SDK Build-Tools;
  + Android SDK Platform;
  + USB Driver;
* Android SDK 30.29;
* SQL Server Management Studio 18.9.1;
* Visual Studio Code 1.57;
* Notepad++ 8.1;
* Postman 8.6.2.

Программное обеспечение сервера:

* Microsoft Visio Professional 2019;
* Microsoft Office 2019;
* Git 2.32;
* .NET Framework Developer Pack 4.8;
* Microsoft Visual Studio 2022 Community, включая следующие компоненты:
  + .NET desktop development Workload;
  + Universal Windows Platform development Workload;
  + Data storage and processing Workload;
  + Entity Framework (EF); IIS, ASP.NET and web development, Mobile development with .NET;
* Android Studio 4.2.1, включая следующие компоненты:
  + Android SDK Tools;
  + Android SDK Platform-Tools;
  + Android SDK Build-Tools;
  + Android SDK Platform;
  + USB Driver;
* Android SDK 30.29;
* SQL Server Management Studio 18.9.1;
* Visual Studio Code 1.57;
* Notepad++ 8.1;
* Postman 8.6.2;
* сервер Gogs;
* IIS Express 10.0, Nginx 1.21, Apache 2.4.48;
* Microsoft SQL Server 2017 Express;

В IT-отделе происходит разработка и тестирование заказного программного обеспечения.

**3 СВЕДЕНИЯ О ПРОХОЖДЕНИИ ПРАКТИКИ В КАЧЕСТВЕ ДУБЛЕРА ИНЖЕНЕРНО-ТЕХНИЧЕСКОГО РАБОТНИКА**

Во время прохождения практики в качестве дублера программиста я выполнял следующие действия:

* ознакомление с системным и прикладным программным обеспечением в организации;
* работа с документацией;
* сбор и структурирование материала по теме индивидуального задания « Разработка эмулятора машины Поста »;
* анализ алгоритмов, в том числе с применением инструментальных средств;
* разработка кода программного продукта на основе готовой спецификации на уровне модуля;
* написание программного кода, реализующего функциональность разрабатываемого приложения;
* разработка интерфейса приложения;
* отладка программного продукта с использованием спе­циализированных программных средств;
* разработка тестовых наборов и тестовых сценариев;
* осуществление рефакторинга и оптимизации программного кода;
* тестирование программного продукта;
* разработка технической документации к программному продукту;
* обслуживание оргтехники;
* установка и настройка необходимого программного обеспечения;
* разработка кода по заданию руководителя практики.

Выполненная работа соответствует профессиональным компетенциям, предусмотренным программой производственной практики ПМ.01 специальности 09.02.07 Информационные системы и программирование.

**4 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ**

**4.1 Постановка задачи**

В период практики необходимо написать мобильное и desktop-приложение эмулирующее работу машины Поста с возможностью добавления и удаления команд, редактированием начальных значений, расположенных на ленте, возможностью пошагово просматривать результат выполнения каждой команды.

Программа должна предлагать следующие возможности:

* вывода ленты на экран;
* ввода начальный значений ленты;
* отображение положения каретки;
* возможность навигации по ленте;
* написание собственного алгоритма;
* пошаговое выполнение алгоритма.

Входными данными будут:

* исходное состояние ленты;
* набор команд для выполнения.

Выходными данными будут:

* результат выполнения заданных команд над исходной лентой.

Также необходимо разработать пакет тестов для функции, вычисляющей значения исследуемой функции.

**4.2 Проектирование структуры программы**

**4.2.1 Desktop версия приложения**

Разработанное приложение имеет следующие окна:

* заставка;
* главное окно, которое содержит области:
* ленты, разделённой на ячейки (поле для ввода входных данных и вывода результата)
* ввода команд;
* навигация по программе;
* справка.

Обращение к окнам реализовано с помощью кнопок (за исключением окна заставки).

**4.2.2 Мобильная версия приложения**

Разработанное приложение имеет следующие окна:

* окно заставки;
* главное окно, которое содержит области:
* ленты, разделённой на ячейки (поле для ввода входных данных и вывода результата)
* ввода команд;
* навигация по программе;
* справка;
* окно добавления команды.

Обращение к окнам реализовано с помощью кнопок (за исключением окна заставки).

**4.3 Функциональное взаимодействие модулей программы**

**4.3.1 Desktop версия приложения**

На основе разработанного технического проекта был написан программный продукт «эмулятор машины Поста». Для создания приложения была использована среда программирования Visual Studio 2022 и язык программирования С#. В программе используются следующие языковые модули:

* System.Windows.Controls - предоставляет классы для создания элементов, известных как элементы управления, позволяющих пользователю взаимодействовать с приложением;
* System.Diagnostics - содержит типы, позволяющие взаимодействовать с системными процессами, журналами событий и счётчиками производительности;
* System - содержит фундаментальные и базовые классы, определяющие часто используемые типы значений и ссылочных данных, события и обработчики событий, интерфейсы, атрибуты и исключения обработки;
* System.Windows - содержит типы, используемые в приложениях WindowsPresentationFoundation (WPF), включая клиенты анимации, элементы управления пользовательского интерфейса, привязку данных и преобразование типов;
* CefSharp - содержит типы, поддерживающие компиляцию и создание кода на языке C#, а также поддерживающие взаимодействие между средой DLR и языком C#;
* System.IO - cодержит типы, поддерживающие ввод и вывод, включая возможности чтения и записи данных в потоках как синхронно, так и асинхронно, реализации пользовательского ведения журнала и обработки входящих и исходящих потоков данных в последовательных портах;
* System.Data - содержит типы для доступа к данным из различных источников и для управления этими данными. Пространство имен верхнего уровня и несколько дочерних пространств имен образуют архитектуру ADO.NET и поставщиков данных ADO.NET [9].
* System.Threading - содержат типы, обеспечивающие возможности многопотокового программирования. Дочернее пространство имен предоставляет типы, которые упрощают задачу написания параллельного и асинхронного кода;
* System.Collections.Generic – работа с обобщёнными коллекциями (List<string>).

В программе были созданы следующие пользовательские модули:

* Carrige.cs – класс каретки, обеспечивающий отслеживание её местоположения и возможности его изменения;
* MainWindow.xaml.cs (главное окно программы) - обеспечивает функциональность для ввода и вывода данных в машину Поста;
* Commands.cs – класс списка команд, подаваемых на выполнение;
* CustomButton.cs - класс переопределения элемента кнопки для использования его в ленте;
* Operation.cs – класс для описания операции и ссылок на другие операции; OperationCarrige.cs – класс для отображения состояния выполнения списка операций;
* Section.cs – класс одной ячейки леты, необходим для отслеживания её состояния;
* Quest.xml.cs - класс окна справки.

В модуле MainWindow.xaml.cs описаны и используются следующие процедуры:

* private MainWindowk() – вызываются основные функции и задаются значения по умолчанию;
* private void Task\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e) – обработчик возможности перемещения окна;
* private void Button\_Click(object sender, RoutedEventArgs e) – обработчик кнопки закрытия окна
* private void Button\_Click\_1(object sender, RoutedEventArgs e)- обработчик кнопки минимизации ока;
* private void btnReset\_MouseDown(object sender, MouseButtonEventArgs e) – обработчик кнопки возвращения ленты в исходное состояние;
* private void btnMinus\_MouseDown(object sender, MouseButtonEventArgs e) – обработчик кнопки удаления последней команды;
* private void btnPlus\_MouseDown(object sender, MouseButtonEventArgs e) – обработчик кнопки добавления новой команды;
* private async void btnStart\_MouseDown(object sender, MouseButtonEventArgs e) – обработчик кнопки старта;
* private void AddTestCommands() – метод добавления команд по умолчанию;
* private void btnNext\_MouseDown(object sender, MouseButtonEventArgs e) – обработчик кнопки выполнения одной команды;
* private void noNameStackScroller\_PreviewMouseWheel(object sender, MouseWheelEventArgs e) – обработчик пролистывания ленты;
* private void RefreshLine() – метод возврата ленты в исходное состояние;
* private void CommandsInit() – метод отображения команд;
* private void ApplyCommands() – метод выполнения одной команды;
* private void LineInit() – метод отображения ленты;
* private void rightSide\_MouseDown(object sender, MouseButtonEventArgs e) – обработчик правой кнопки навигации по ленте;
* private void leftSide\_MouseDown(object sender, MouseButtonEventArgs e) - обработчик левой кнопки навигации по ленте;
* private void leftSide\_Copy\_MouseDown(object sender, MouseButtonEventArgs e) – обработчик центральной кнопки навигации по ленте;

В модуле addCommand.xaml.cs описаны и используются следующие процедуры:

* public addCommand() – инициализация окна;
* private void TextBox\_PreviewTextInput(object sender, TextCompositionEventArgs e) – обработчик ввода в поля ввода
* private void Button\_Click(object sender, RoutedEventArgs e) – обработчик кнопки отмены;
* private void Button\_Click\_1(object sender, RoutedEventArgs e) – обработчик кнопки отправить;
* private void ComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e) – обработчик выбора команды;

В модуле Carrige.cs описаны и используются следующие процедуры:

* public Carriage(int locationIndex) – конструктор класса каретки;
* public void CarriageRight() – сдвиг каретки вправо по ленте на 1;
* public void CarriageLeft() – сдвиг каретки в влево по ленте на 1;
* public void SetNewLocation(int index) – метод установки значения метоположения каретки;
* public int GetCarriageLocation() – метод получения текущего положения каретки.

В модуле Quest.xaml.cs описаны и используются следующие процедуры:

* public quest() – конструктор класса;
* private void Button\_Click(object sender, RoutedEventArgs e)- обработчик нажатия кнопки назад.

В модуле Commands.cs описаны и используются следующие процедуры:

* public List<Operation> GetCommands() – метод получения списка команд;
* private List<Operation> commandsList – конструктор класса;
* public void AddCommand(int inputOperationId, int inputLink) – метод добавления команды;
* public void AddCommand(int inputOperationId, int inputLink, int inputLink2) – метод добавления команды;
* public void RemoveLastCommand() – очистка списка команд.

В модуле CustomButton.cs описаны и используются следующие процедуры:

* public CustomButton(int id,bool cheked) – конструктор класса;

В модуле Operation.cs описаны и используются следующие процедуры:

* public Operation(int inputIndex, int inputCommandId, int inputLink) – конструктор класса;
* public Operation(int inputIndex, int inputCommandId, int inputLink, int inputLink2) – конструктор класса;
* public string GetOperationIcon() – метод получения иконки операции;
* public int GetOperationId() – метод получения id операции;
* public int GetLink() – метод получения ссылки на другую операцию;
* public int GetLink2() – метод получения ссылки на другую операцию.

В модуле Section.cs описаны и используются следующие процедуры:

* public Section(int inputIndex) – конструктор класса;
* public void Check() – метод изменения состояния ячейки на противоположное;
* public void SetCheck() - метод изменения состояния ячейки на закрашенную;
* public void SetUncheck() - метод изменения состояния ячейки на не закрашенную;
* public bool GetChecked() – метод получения состояния ячейки;
* public int GetIndex() – метод получения индекса ячейки.

**4.3.2 Мобильная версия приложения**

На основе разработанного технического проекта была написана мобильная версия программного продукта «эмулятора машины Поста». Для создания приложения была использована среда программирования Android Studio и язык программирования Kotlin. В программе используются следующие языковые модули:

* android.content.Context – это базовый абстрактный класс, реализация которого обеспечивается системой Android. Этот класс имеет методы для доступа к специфичным для конкретного приложения ресурсам и классам и служит для выполнения операций на уровне приложения, таких, как запуск активностей, отправка широковещательных сообщений, получение намерений и прочее;
* android.graphics.Canvas – предоставляет методы для рисования, которые отображают графические примитивы на исходном растровом изображении;
* android.graphics.Color – содержит несколько констант и методов для конвертации и извлечения цветового компонента;
* android.graphics.Paint – содержит стили, цвета и другую графическую информацию для рисования графических объектов;
* android.view.View – компонент(виджет), который рисуется на экране;
* java.util.ArrayList – класс, позволяющий создавать списки;
* android.app.AlertDialog – это уведомление, которое появляется, когда определенное приложение использует систему Android для размещения важной информации для пользователя;
* android.content.Intent – класс, объекты которого используются для взаимодейтвия между отдельными частями Android приложения, либо между различными приложениями системы;
* android.os.Bundle – необходим для временного хранения данных в процессе выполнения;
* android.view.Menu – класс, представляющий меню в приложениях;
* android.view.MenuItem – класс, отвечающий за пункты меню;
* android.widget.Toast – это небольшое, всплывающее сообщение, появляющееся внизу и экрана на непродолжительное время;
* androidx.appcompat.app.AppCompatActivity – это базовый класс для действий, которые используют функциональные возможности панели поддержки библиотеки;
* android.widget.EditText – это стандартный виджет ввода текста в приложениях Android;
* java.io.FileOutputStream – поток вывода, который содержит методы, записывающие данные в файл;
* android.os.AsyncTask – предлагает простой и удобный механизм для перемещения трудоёмких операций в фоновый поток.

В программе были созданы следующие пользовательские модули:

* Carrige.kt – класс каретки, обеспечивающий отслеживание её местоположения и возможности его изменения;
* MainActivity.kt (главное окно программы) - обеспечивает функциональность для ввода и вывода данных в машину Поста;
* Commands.kt – класс списка команд, подаваемых на выполнение;
* Operation.kt – класс для описания операции и ссылок на другие операции;
* OperationCarrige.kt – класс для отображения состояния выполнения списка операций;
* Section.kt – класс одной ячейки леты, необходим для отслеживания её состояния.
* Splash.kt – окно заставки приложения

В модуле MainActivity.kt описаны и используются следующие процедуры:

* class MainActivity – класс основного окна;
* private fun applyCommands() – метод для выполнения команды;
* fun lineInit() – метод инициализации и отрисовки ленты ;
* fun commandsIniit() – метод отрисовки списка команд;
* private fun addTestCommands() – метод добавления команд по умолчанию;
* private fun refreshLine() – метод возвращения леты в исходное положение;
* private fun carriageUpdate() – метод обновления положения каретки;
* private fun addListeners() – метод добавления обработчиков на элементы управления;
* private fun operationCarriageUpdate() – метод обновления состояния выполнения команд;
* private fun playCommands() – метод нажатия на кнопку автовыполнения команд;
* private fun navigationBottomButtonsListeners() – метод добавления обработчиков на нижнюю панель управления;
* private fun setDefaulScroll() – метод установки значения scroll по центру для ScrollView;
* private fun navigationCarriageListeners() – метод добавления обработчиков для элементов навигации по ленте.

В модуле Carrige.kt описаны и используются следующие процедуры:

* fun carriageRight() – метод для смещения каретки на одну ячейку вправо;
* fun carriageLeft() – метод для смещения каретки на одну ячейку влево;
* fun setNewLocation(index:Int)- метод установки местоположения каретки;
* getCarriageLocation() – метод получения положения каретки;
* getLastCarriageLocation()- метод получения списка последних позиций каретки;
* fun clearCarrigeList() – метод очистки списка последних позиций каретки.

В модуле Commands.kt описаны и используются следующие процедуры:

* fun getCommands() – метод для получения списка команд;
* fun addCommand(inputOperationId:Int,inputLink:Int)- метод добавления новой команды по id команды и ссылке на следующую после неё команду;
* addCommand(inputOperationId:Int,inputLink:Int,inputLink2: Int)метод добавления новой команды по id команды и двум ссылкам на следующие команды;
* fun removeLastCommand() – метод очистки списка команд.

В модуле Operation.kt описаны и используются следующие процедуры:

* constructor(inputIndex:Int,inputCommandId:Int,inputLink:Int) – конструктор объекта Operation по заданным id, id команды и ссылке на другую команду;
* constructor(inputIndex:Int,inputCommandId:Int,inputLink:Int,inputLink2:Int) - конструктор объекта Operation по заданным id, id команды и двум ссылкам на другие команды;
* fun getOperationIcon() - метод получения иконки операции по её id;
* fun getOperationId() – метод получения id операции;
* fun getLink() – метод получения ссылки на команду;
* fun getLink2() – метод получения ссылки на команду.

В модуле Section.kt описаны и используются следующие процедуры:

* fun check() – метод для присваивания противоположного состояния ячейки;
* fun setCheck() – метод установки состояния ячейки на закрашенную;
* fun setUncheck() – метод установки состояния ячейки на не закрашенную;
* fun getChecked() – метод возвращающий значение состояния ячейки;
* fun getIndex() - метод получения индекса ячейки.

**4.4 Разработка тестов**

Для осуществления корректной работы разработанного приложения, в части правильности выполнения операцию удаления команды, был разработан набор тестов. Данные тест охватывает все возможные тестовые случаи, которые необходимо проверить: возможность удаления последней операции при нажатии на соответствующую кнопку при пустом и обычном наборе команд.

Аннотация тестов:

|  |  |
| --- | --- |
| Название проекта | Эмулятор машины Поста |
| Рабочая версия | 1.0 |
| Имя тестирующего | Панычев Н. О. |
| Дата проведения теста | 13.12.2023 |

Сначала проверим возможность удаления последней операции:

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_1 |
| Приоритет тестирования | Высокий. |
| Заголовок/название теста | Удаление из непустого списка. |
| Краткое изложение теста | Проверка удаления последней команды из списка с несколькими элементами. |
| Этапы теста | * создание списка с несколькими командами; * нажатие на кнопку удаления последней команды. |
| Тестовые данные | Непустой список команд - [1,2,3]. |
| Ожидаемый результат | Количество элементов в списке должно уменьшиться на 1 после удаления. |
| Фактический результат | Количество элементов в списке уменьшилось на 1 после удаления. |
| Статус | Удачный. |
| Предварительное условие | Список команд с несколькими элементами. |
| Постусловие | Количество элементов в списке должно уменьшиться на 1 после удаления. |
| Примечания/комментарии | Этот тест проверяет функцию удаления последней команды при наличии нескольких элементов в списке команд. Цель состоит в том, чтобы убедиться, что функция удаляет последний элемент корректно и правильно обновляет список. При успешном выполнении этого теста ожидается уменьшение количества элементов в списке на 1 после вызова функции удаления последней команды. Этот тест помогает гарантировать корректность удаления элементов из списка команд в общем случае. |

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_2 |
| Приоритет тестирования | Средний. |
| Заголовок/название теста | Удаление из пустого списка. |
| Краткое изложение теста | Проверка удаления последней команды из пустого списка команд. |
| Этапы теста | * удаление всех команд; * нажатие на кнопку удаления последней команды. |
| Тестовые данные | Пустой список команд - []. |
| Ожидаемый результат | Список остаётся пустым. |
| Фактический результат | Список остаётся пустым. |
| Статус | Удачный. |
| Предварительное условие | Список команд пуст. |
| Постусловие | Список остался пустым после удаления. |
| Примечания/комментарии | Этот тест проверяет поведение программы при попытке удаления элемента из пустого списка команд. Целью является подтверждение корректного обработки такой ситуации функцией: ожидается, что список останется пустым после попытки удаления элемента, и количество элементов в списке останется равным 0. Этот тест важен для проверки безопасности функции при работе с пустыми данными и обеспечения корректного поведения в этом случае. |

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_3 |
| Приоритет тестирования | Высокий. |
| Заголовок/название теста | Удаление последнего элемента из списка с одним элементом. |
| Краткое изложение теста | Проверка удаления единственного элемента списка команд. |
| Этапы теста | * создание списка команд с одним элементом; * нажатие на кнопку удаления последней команды. |
| Тестовые данные | Список с одним элементом - [1]. |
| Ожидаемый результат | Список становится пустым. |
| Фактический результат | Список становится пустым. |
| Статус | Удачный. |
| Предварительное условие | Список команд с одним элементом. |
| Постусловие | Список стал пустым после удаления. |
| Примечания/комментарии | Этот тест проверяет поведение программы при удалении единственного элемента из списка команд. Цель состоит в том, чтобы убедиться, что функция корректно обрабатывает сценарий удаления последнего элемента из списка, оставляя его пустым после выполнения операции. Ожидается, что после вызова функции у списка не останется элементов, и его размер будет равен 0. Этот тест важен для проверки специального случая, когда список содержит только один элемент. |

|  |  |
| --- | --- |
| Тестовый пример # | TC\_UI\_4 |
| Приоритет тестирования | Средний. |
| Заголовок/название теста | Производительность. |
| Краткое изложение теста | Проверка времени выполнения функции для больших списков. |
| Этапы теста | * создание большого списка команд; * Замер времени выполнения функции удаления последней команды. |
| Тестовые данные | Большой список команд. |
| Ожидаемый результат | Функция выполняется эффективно для больших списков. |
| Фактический результат | Функция выполняется эффективно для больших списков. |
| Статус | Удачный. |
| Предварительное условие | Большой список команд. |
| Постусловие | Функция работает быстро для большого объёма данных. |
| Примечания/комментарии | Этот тест направлен на оценку времени выполнения функции удаления последней команды при работе с большим объемом данных. Целью является убеждение в эффективности работы функции даже при больших списках команд. Ожидается, что функция будет работать быстро и эффективно, несмотря на увеличение объема данных. Замеряется время выполнения функции для большого списка команд, и результат сравнивается с ожидаемым производительным поведением. Этот тест важен для подтверждения хорошей производительности функции при реальной нагрузке большого объема данных. |

**4.5 Работа с приложением**

**4.5.1 Desktop версия приложения**

При запуске программы открывается заставка (рисунок 1).

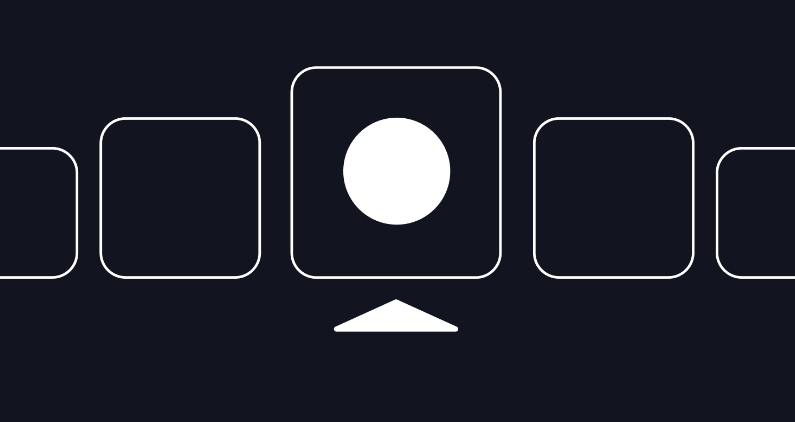


Рисунок 1 – Заставка

После завершения загрузки, откроется Главное окно программы.

Интерфейс программы имеет следующие элементы (рисунок 2):

* поле команд, в нём отображаются все введённый вами команды (1);
* кнопка сброса состояния ленты (2);
* кнопка быстрого выполнения команд (3);
* кнопка выполнения одной команды (4);
* кнопка удаления последней команды (5);
* кнопка добавления команды (6);
* кнопка, помогающая пролистать ленту до левого края (7);
* кнопка, помогающая пролистать ленту к центру (8);
* кнопка, помогающая пролистать ленту до правого края (9).

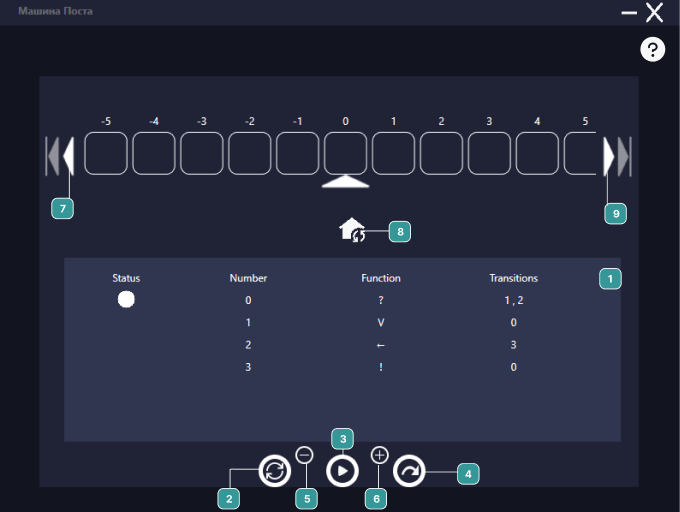


Рисунок 2 – Основное окно программы

Чтобы написать свой алгоритм вы можете удалить команды по умолчанию из поля команд, затем добавить свои нажатием на кнопку «+», после чего выбрать необходимую команду и нажать на кнопку создать. Чтобы выполнить свой алгоритм моментально вы можете нажать на центральную кнопку внизу, если же вы хотите пролистывать шаг за шагом, то вы можете нажать на кнопку правее с изображением стрелочки.

Если у пользователя возникнуть вопросы по работе с программой, вы можете вызвать справочное окно, нажав на кнопку «?» в правом верхнем углу экрана (рисунок 3).



Рисунок 3 – Окно справки

**4.5.2 Мобильная версия приложения**

При запуске программы открывается заставка (рисунок 5).

После завершения загрузки, откроется главное окно программы (рисунок 6).

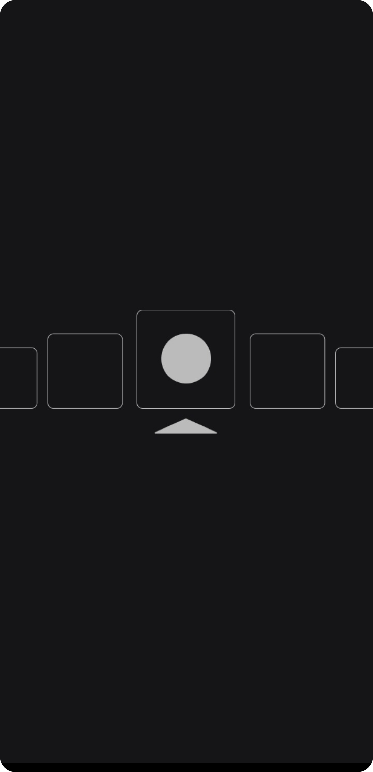


Рисунок 5 – Заставка

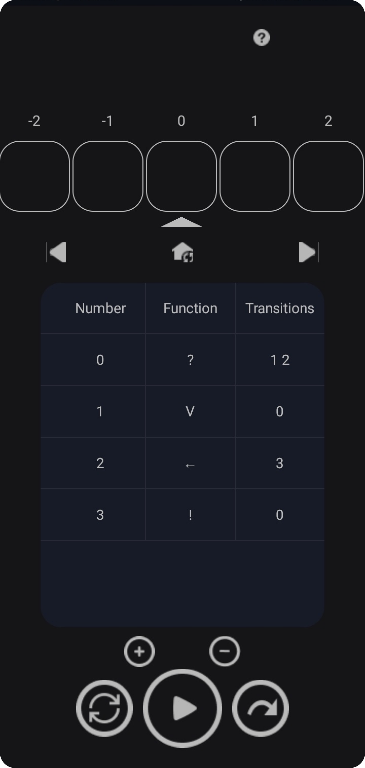


Рисунок 6 – Основное окно программы

Чтобы написать свой алгоритм вы можете удалить команды по умолчанию из поля команд, затем добавить свои нажатием на кнопку «+», после чего откроется окно добавления команды (рисунок 7), после чего выбрать необходимую команду и нажать на кнопку создать. Чтобы включить автоматическое выполнение команд вы можете нажать на центральную кнопку внизу, если же вы хотите пролистывать шаг за шагом, то вы можете нажать на кнопку правее с изображением стрелочки.

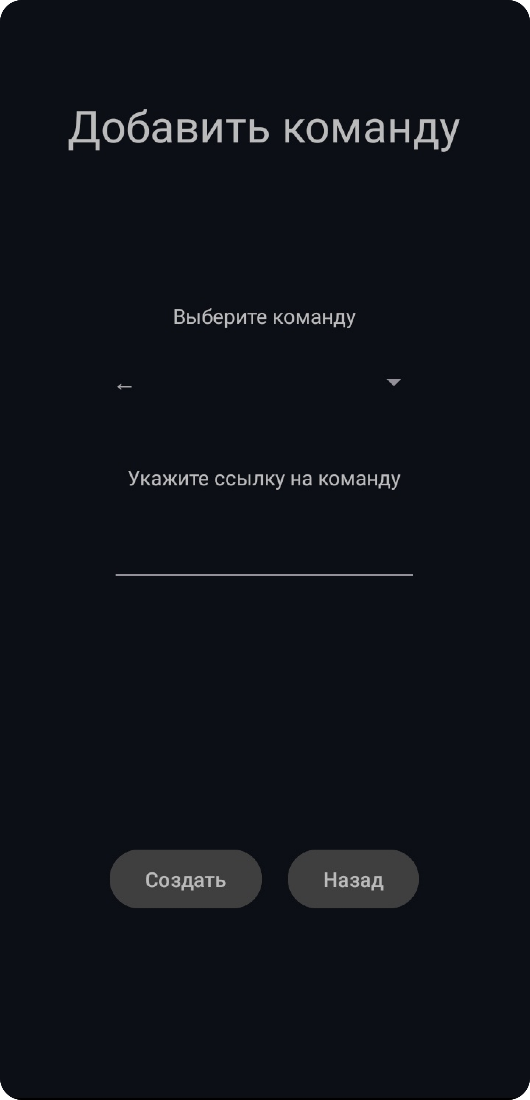


Рисунок 7 – Окно добавления новой команды

Если вы хотите узнать дополнительную информацию о том что такое машина Поста, то вы можете вызвать справочное окно, нажав на кнопку «?» в правом верхнем углу главного окна (рисунок 9).

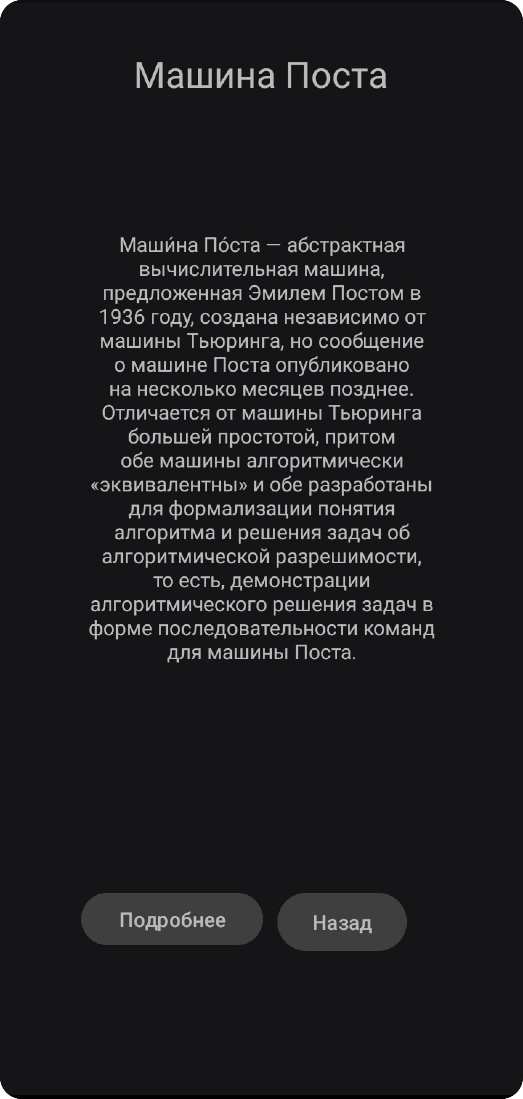


Рисунок 8 – Окно справки

**ЗАКЛЮЧЕНИЕ**

В результате прохождения производственной практики мною были закреплены теоретические знания, полученные при изучении ПМ.01; отработаны профессиональные компетенции по получаемой специальности; приобретены практические навыки в будущей профессиональной деятельности; выполнено индивидуальное задание.

Мною было разработано приложение эмулятор машины Поста.

Программа обеспечивает следующие возможности:

* вывода ленты на экран;
* ввода начальных значений ленты;
* отображение положения каретки;
* возможность навигации по ленте;
* написание собственного алгоритма;
* пошаговое выполнение алгоритма.

В ходе выполнения разработки данного приложения были:

* проанализирована работа машины Поста;
* спроектирована структура приложения и программного кода;
* разработаны справочные материалы;
* выбраны средства реализации приложения;
* разработан программный код приложения;
* разработан оптимальный набор тестов.

Основными достоинствами программы являются:

* программа проста в установке;
* данный продукт полезен для развития логического мышления и повышения уровня умений в построении алгоритмов;
* наличие справочной системы.

К недостаткам данной программы можно отнести то, что в ней тяжело разобраться без прочтения руководства.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Албахари, Б. С# 7.0. Справочник. Полное описание языка /   
   Б. Албахари, Дж. Албахари. – Москва: Вильямс, 2018. – 1024 с.
2. Виссер, Д. Разработка обслуживаемых программ на языке С# /   
   Д. Виссер. – Москва: ДМК Пресс, 2019. – 194с.
3. Дарвин, Я. Ф. Android. Сборник рецептов: задачи и решения для разработчиков приложений / Я. Ф. Дарвин. – 2-е изд., пер. с англ. – Санкт-Петербург: ООО «Альфа-книга», 2018. – 768 с.
4. Джепикс, Ф. Язык программирования C# 7 и платформы .NET и .NET Core / Ф. Джепикс, Э. Троелсен. – Москва: Вильямс, 2018. – 1328 с.
5. Мюллер, Д. П. С# Для чайников / Д. П. Мюллер, Б. Семпф, Ч. Сфер. – Москва: Диалектика, 2019. – 608 с.
6. Скит, Д. С# Для профессионалов. Тонкости программирования /   
   Д. Скит. – Москва: Вильямс, 2019. – 608 с.
7. Официальная документация Microsoft для работы с языком программирования C# // Microsoft.com: [сайт]. – URL: https://docs.microsoft.com/ru-ru/documentation (дата обращения: 01.12.2023).
8. Библиотека классов .NET Framework // Microsoft.com: [сайт]. – URL: https:// msdn.microsoft.com/ru-ru/library/mt472912(v=vs.110).aspx (дата обращения: 01.12.2023).
9. Как пользоваться timer в c# // Microsoft.com: [сайт]. – URL: https://docs.microsoft.com/ru-ru/dotnet/api/system.timers.timer?view=netcore-2.1 (дата обращения: 01.12.2023).
10. Общие сведения о платформе Android // Developers: [сайт]. – URL: https://developer.android.com/guide (дата обращения: 30.11.2023).