ВВЕДЕНИЕ

Важным элементом процесса подготовки специалиста в области программирования является практика, итоговым документом которой будет считаться отчет по производственной практике программиста.

Современное общество характеризуется резким увеличением объемов и потоков информации, требуется создание больших баз данных, расширение связи между предприятиями, их структурными подразделениями и отраслями. Компьютер и интернет прочно вошли в нашу жизнь, создаются новые программы, усложняются сами вычислительные машины, и профессия программиста стала одной из самых востребованных.

10.04.2023

Тема:Инструктаж о прохождении практики. Знакомство с программой практики и порядок ее проведения, изучения правил внутреннего распорядка, знакомство с графиком работы студентов, введения дневника практики, составление отчета. Инструктаж по технике безопасности, пожаробезопасности, производственной санитарии под роспись в журнале. Правила безопасности при работе с компьютером

Я прохожу практику в ГБУ РД «Хас ЦРБ»Кокрекская УБ.

Рисунок 1-Организация, в которой проходит практика

Мой руководитель по производственной практике провел инструктаж по технике безопасности:

1.1. Соблюдать Устав, правила внутреннего распорядка обучающихся, требования инструкции по охране труда и иные локальные акты колледжа.

1.2. Своевременно и качественно выполнять распоряжения администрации предприятия, наставника на месте практики, руководителя практики от колледжа, воздерживаться от действий, мешающих нормальной деятельности предприятия.

1.3. Соблюдать требования по охране труда, технике безопасности, производственной санитарии, гигиене труда и противопожарной безопасности, предусмотренными правилами и инструкциями на месте практики.

1.4. Соблюдать чистоту, порядок в помещении по месту практики и на прилегающей территории.

После мне предоставили рабочее место с компьютером.

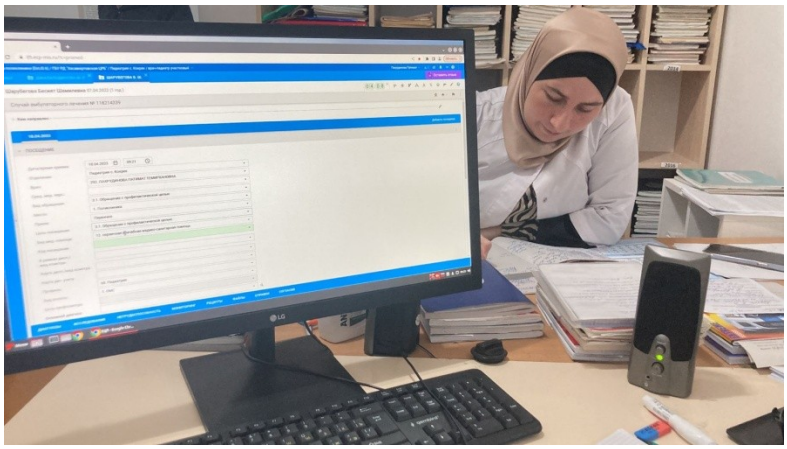


Рисунок 2 -Рабочее место

Технические характеристикикомпьютера:

1. Диагональ экрана – 21.5 дюйм;
2. Разрешение экрана – Full HD (1920x1080), 60 Гц;
3. Процессор – Intel Core i3 10110U;
4. Размер оперативной памяти – 4 ГБ;
5. Объем жесткого диска – 256 ГБ;
6. Тип системы –64-разрядная операционная система.

В данном компьютере установлена профессиональная операционная система – Windows10.

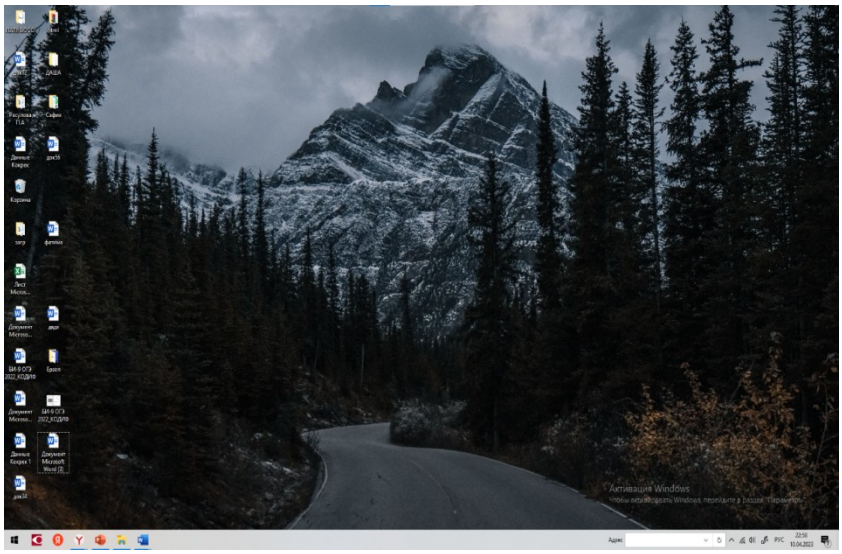


Рисунок 3 –Рабочий стол компьютера

11.04.2023

Тема: Концепция разработки программного модуля

Модульное программирование – это организация программы как совокупность независимых блоков, называемых модулями, структура и поведение которых подчиняются определенным правилам.

Порядок разработки программного модуля.

1. изучение и проверка спецификации модуля, выбор языка программирования; (т.е. разработчик, изучая спецификацию, выясняет, понятна она ему или нет, достаточно ли полно она описывает модуль; затем он выбирает язык программирования, на котором будет написан модуль, хотя язык программирования может быть единым для всего ПС);
2. выбор алгоритма и структуры данных (здесь выясняется не известны ли какие-либо алгоритмы для решения поставленной задачи и если есть, то воспользоваться им);
3. программирование модуля (написание кода программы);
4. шлифовка текста модуля (редактирование имеющихся комментариев, добавление дополнительных комментариев, для того чтобы обеспечить требуемое качество);
5. проверка модуля (проверяется логика работы модуля, отлаживается его работа);
6. компиляция модуля.

За последнее время применение компьютеров в медицине чрезвычайно повысилось. Практическая медицина становится все более и более автоматизированной. Сложные современные исследования в медицине немыслимы без применения вычислительной техники.Количество информации, которое получается при таких исследования так огромно, что без компьютера человек был бы неспособен ее воспринять и обработать. И поэтому я решила создать программу для больницы в которой я прохожу практику.

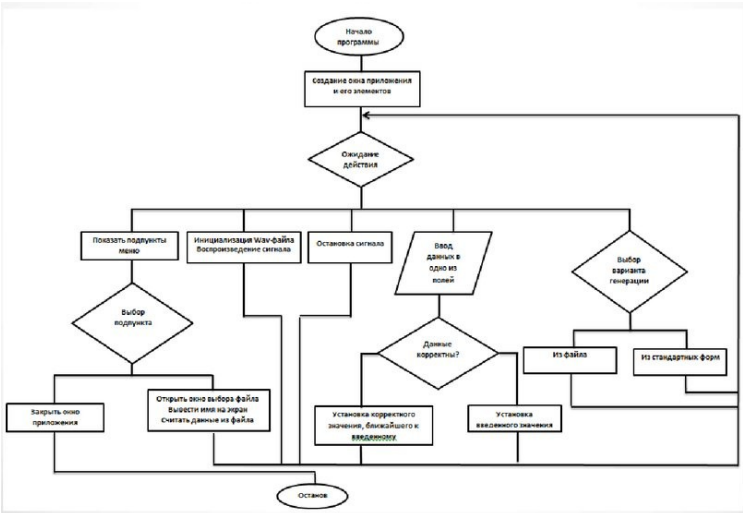


Рисунок 4 – Алгоритм работы программы

Основной задачей, данной работы, является разработка автоматизированной базы данных для больницы, которая поможет пользователю легко найти нужную информацию о любом сотруднике или пациенте.

Основные задачи, решаемые системой:

- ведение амбулаторной карты и истории болезни пациента в электронном виде;
- отражение процесса оказания диагностической и медицинской помощи в электронной медицинской карте пациента, которая объединяет в себе историю болезни и амбулаторную карту;
- электронная карта представляет собой хорошо организованное, структурированное хранилище информации;
- в системе предусмотрены удобные механизмы ввода и просмотра данных, повышающие скорость работы, удобство ее поиска и наглядность представления информации;
- возможность редактировать и добавлять новые данные.

12.04.2023

Тема:Разработка кода программного продукта на основе готовой спецификации на уровне модуля

Программа «Учёт пациентов в регистратуре поликлиники» для учёта пациентов в регистратуре поликлиники.

1) Для добавления нового пациента необходимо ввести: Ф.И.О., пол, дату рождения, дату поступления, текущее заболевание и выбрать из списка лечащего врача.

Программа должна обеспечивать выполнение следующих функций:

- добавление пациента;
- вывод списка пациентов (в табличной форме);
- удаление пациента на случай, когда он выздоровеет либо умрет;
- добавление лечащего врача, процедур, лекарств.

2) Составить UML-диаграммы: вариантов использования, классов, последовательности.

3) Провести тестирование на правильность ввода данных и правильность обработки исключительных ситуаций.

Для работы с базой данных используется класс sqliteclass, в котором содержатся все необходимые конструкции работы с базой.

Для написания программы, задан язык C#.

Microsoft Visual Studio — линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Microsoft Windows, Windows Mobile, WindowsCE, .NETFramework, .NETCompactFramework и MicrosoftSilverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода.

Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения. Visual Studio позволяет создавать и подключать сторонние дополнениядля расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода, добавление новых наборов инструментов(например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования или инструментовдля прочих аспектов цикла разработки программного обеспечения.

В таблице 1 приведено описание методов классов программы.

Таблица 1- Классыпрограмм

Имя класса	Название метода	Описание метода
Path	Combine	Указание полного пути к файлу базы данных
mydb	drExecute	Выполнение запроса к базе с возвращением результата
	iExecuteNonQuery	Выполнение запроса на добавление или удаление без возврата

Описание файлов, используемых в проекте, представлено в таблице 2.

Таблица 2 - Файлы используемые в проекте

Имя файла	Описание файла
Form1.cs	Содержит описание класса Form1
Form1.Designer.cs	Содержит описание класса дизайна формы Form1
Form2.cs	Содержит описание класса Form2
Form2.Designer.cs	Содержит описание класса дизайна формы Form2
Form3.cs	Содержит описание класса Form3
Form3.Designer.cs	Содержит описание класса дизайна формы Form3
Form4.cs	Содержит описание класса Form4
Form4.Designer.cs	Содержит описание класса дизайна формы Form4
Form5.cs	Содержит описание класса Form5
Form5.Designer.cs	Содержит описание класса дизайна формы Form5
Program.cs	Содержит главную точку входа для приложения

13.04.23

Тема:Использование инструментальных средств на этапе отладки программного продукта

Отладкой называют процесс устранения в программе ошибок, которые были найдены на этапе тестирования.

Первая задача, решаемая при отладке — это локализация ошибки, т. е. выявление места в программе, где она была допущена. Степень локализации варьируется от всей программы до модулей, подпрограмм и, наконец, до конкретных операторов, содержащих ошибку.

Вторая задача — это исправление локализованной ошибки.

Принципы работы отладчика.

Отладчик - основное инструментальное средство для отладки программ. Он служит для поиска ошибок самого разного вида путем прогона программы в особом, отладочном, режиме. Отладочный режим отличается от обычного тем, что в нем выполнение программы можно приостанавливать и продолжать в любом месте программы и в любой момент, выполнять программу по шагам, смотреть состояние ячейки памяти или регистра. При указании собрать программу для выполнения в отладочном режиме компилятор добавляет в ее бинарный файл информацию, позволяющую отладчику оперировать с объектами данных и процедурами программы по их именам в исходном тексте.

Отладчик может быть как независимой утилитой (например, GNU GDB), так и частью интегрированной среды разработки программ (Microsoft Visual C++). Базовый набор функций у отладчиков приблизительно одинаков. Отладчики позволяют:

- инициировать выполнение программы в отладочном режиме;
- останавливать выполнение программы;
- приостанавливать и продолжать выполнение программы;
- производить пошаговое выполнение;

- просматривать и изменять значения переменных;
- просматривать и изменять значения ячеек памяти;
- просматривать стек вызовов подпрограмм;
- устанавливать и убирать точки останова (англ, breakpoint) по строкам исходного текста программы;
- устанавливать и убирать точки останова по командам процессора в выполняемой программе.

Выполнение программы в отладочном режиме может приостанавливаться в следующих случаях:

- по команде пользователя;
- при достижении в отлаживаемой программе команды завершения выполнения;
- при генерации процессором прерывания, которое сигнализирует о возникновении события или ошибке некоторого вида;
- при достижении точки останова.

Некоторые виды ошибок в процессе отладки могут быть обнаружены благодаря тому, что процессор при их обнаружении может сгенерировать аппаратное прерывание. Конкретный набор обнаруживаемых таким образом ошибок зависит от архитектуры процессора. Обычно в такой набор входят следующие виды ошибок:

- переполнение стека;
- ошибка доступа к памяти;
- целочисленное деление на нуль;
- неверный код операции.

Пошаговое выполнение программы может производиться с разной степенью детализации:

- по строкам исходного текста программы без захода в вызываемые подпрограммы;
- по строкам исходного текста программы с заходом в вызываемые подпрограммы;
- по командам процессора в выполняемой программе.

14.04.23

Тема:Проведения тестирования программного модуля по определенному сценарию

После разработки программы необходимо провести тестированиеприложения.

Чтобы запустить программу, необходимо открыть файл приложения pacient.exe.

Главное окно показано на следующем скриншоте и представляет собой форму с 4 кнопками:

- Карточка больного;
- Врачи;
- Процедуры;
- Лекарства.

Рисунок 5 - Карточка больного

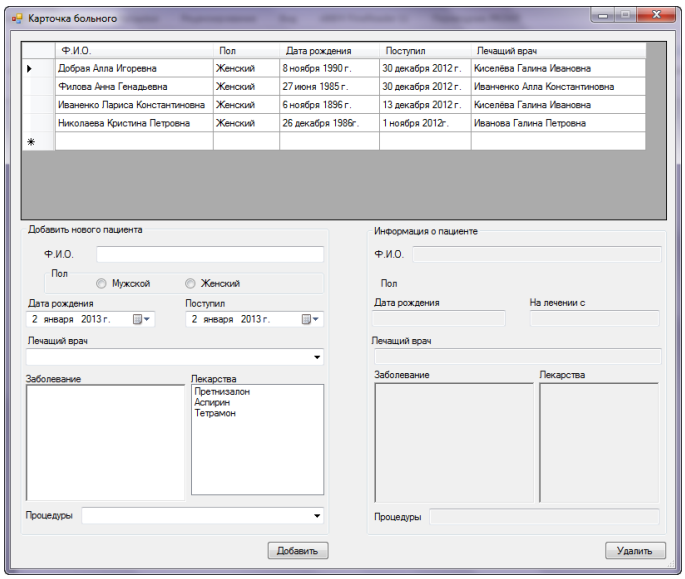


Рисунок 6 – Открытый вид карточки

Здесь представлены следующие элементы:

- Таблица, в которой содержатся все занесенные в БД пациенты.
- Форма добавления нового пациента и его основной информации: ФИО, дата рождения, заболевание и другое.
- Форма просмотра текущей информации о пациенте.

Для того, чтобы удалить запись из БД, необходимо выбрать его из таблицы вверху формы и нажать кнопку «Удалить».

Внимание! Необходимо корректно заполнять и удалять поля, иначе произойдет ошибка:

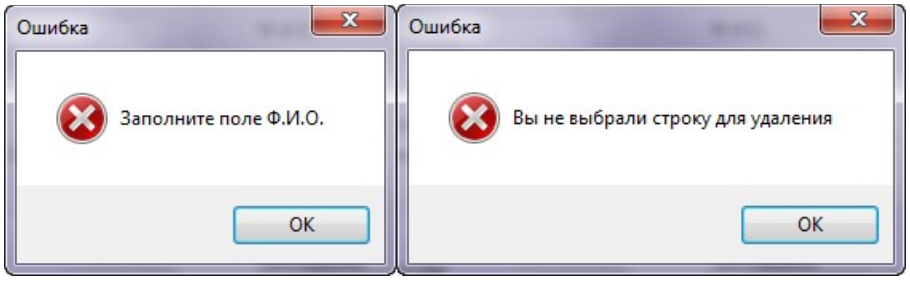


Рисунок 7 – Ошибки

Здесь содержится форма добавления новой записи, таблица с занесенными в БД записями о врачах и форма для их удаления из БД.

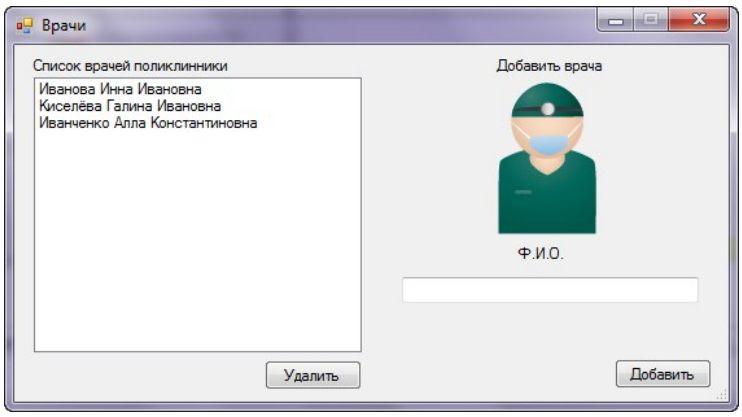


Рисунок 8 - Врачи

Здесь содержится информация о назначенных процедурах, можно добавлять и удалять процедуру.

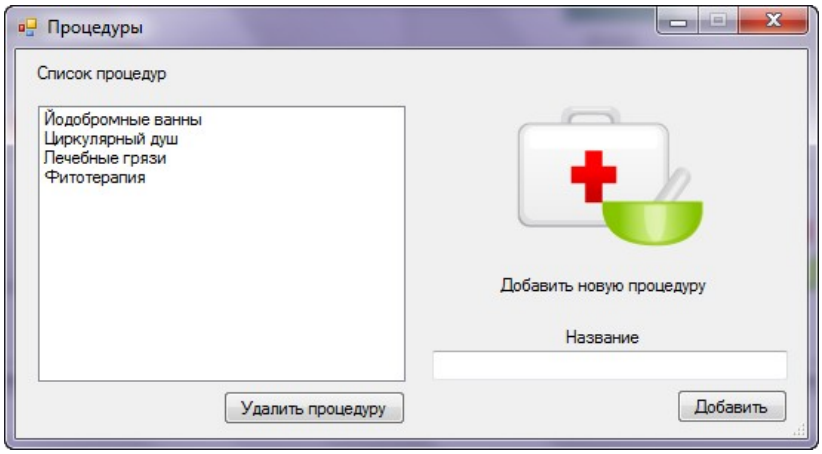


Рисунок 9 - Процедуры

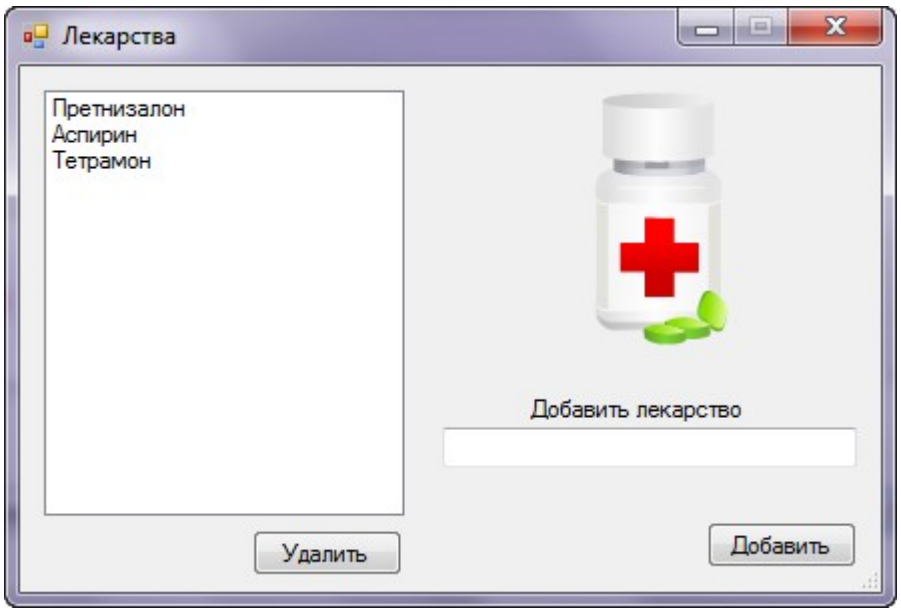


Рисунок 10 – Лекарства

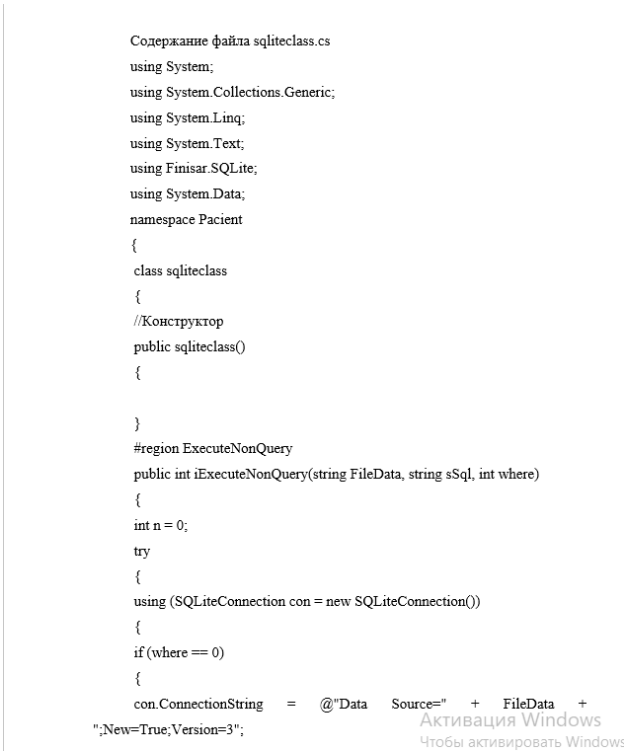
В результате выполнения производственной практики был разработан программный продукт на языке C#. Интерфейс представляет собой экранные форму с меню и диалоговыми окнами.

Плюсами спроектированной программы являются:

1) удобный, интуитивно понятный интерфейс;
2) защита от неправильного ввода;

Также в ходе выполнения работы было проведено тестирование, показывающее, что в программу необходимо добавить проверку на правильность вводимых данных. Найденные ошибки устранены, обработка исключений работает.

Таким образом, в данной производственной работе решена поставленная задача.



```
Содержание файла sqliteclass.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Finisar.SQLite;
using System.Data;
namespace Pacient
{
 class sqliteclass
 {
 //Конструктор
 public sqliteclass()
 {

 }
 #region ExecuteNonQuery
 public int iExecuteNonQuery(string FileData, string sSql, int where)
 {
 int n = 0;
 try
 {
 using (SQLiteConnection con = new SQLiteConnection())
 {
 if (where == 0)
 {
 con.ConnectionString = @"Data Source=" + FileData +
";New=True;Version=3";
```


Рисунок 11 – Начало кода

```
            catch
            {
            MessageBox.Show("Вы не выбрали строку для удаления", "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            }
            }
           }
```

Рисунок 12- Конец кода

Так как код программы очень большой, я решила вставить только начало и конец.

15.04.23

Тема:Разработка эксплуатационной документации

Эксплуатационная документация – это вид технической документации, описывающий порядок установки, настройки и использования разработанного программного обеспечения (автоматизированной системы).

Как правило, документы данного вида являются составляют либо дополнение к "ядру" комплекта технической документации (проектной документации), либо полностью независимы и автономны, а в некоторых случаях являются единственной частью документации на программный продукт. Именно эксплуатационную документацию мы чаще всего подразумеваем, когда говорим о программной документации в целом.

Основные документы, которые входят в состав эксплуатационной документации:

- руководство администратора;
- руководство программиста;

- руководство пользователя;
- руководство оператора.

Руководство администратора.

Для работы данной программы особых драйверов и ПО не нужны. Для программы нужны стандартные ОС драйвера, драйвера для принтера и конечно же сам принтер. Проблем во время эксплуатации быть не должно программа прошла качественные тесты.

Руководство программиста.

Среда Delphi состоит из нескольких отдельно расположенных окон. Используются важные компоненты такие как Form (дизайнер форм), unit (окно редактора), палитра компонентов там вы выбираете нужные вам объекты, которые будут помещены на окно form, инспектор объектов, который позволит вам определить свойство какого-либо объекта, помещенного на форму.

События, использованные в данной программе OnClick, AutoSkroll, BorderStyle, HelpFile, HorzScrollBar, Position, OnActivate, OnHelp, OnShortCut.

Руководство пользователя.

Копируйте папку с программой на любой диск вашего ПК и запускайте фал Scenter.exe. Данные вводятся с клавиатуры и при помощи мыши. Данные сохраняются автоматически после нажатия клавиши “Готово”. Меры безопасности для данных следить чтобы за работающим ПК не было вирусов и других вредоносных программ. Особенности работы с данной программой — это то, что программа легко вас понимает, вам не надо долго учиться чтобы работать в данной программе.

Руководство оператора

Нормативной базой для составления данного документа может являться ГОСТ 19.505-79. ЕСПД. Руководство оператора. Требования к содержанию и оформлению, в котором выделяются следующие разделы:

- назначение программы;
- условия выполнения программы;

- выполнение программы;
- сообщения оператору.

17.04.23

Тема: Проведения тестирования и заполнения протокола тестирования

Модульное тестирование – это тип тестирования программного обеспечения, при котором тестируются отдельные модули или компоненты программного обеспечения. Его цель заключается в том, чтобы проверить, что каждая единица программного кода работает должным образом. Данный вид тестирование выполняется разработчиками на этапе кодирования приложения. Модульные тесты изолируют часть кода и проверяют его работоспособность. Единицей для измерения может служить отдельная функция, метод, процедура, модуль или объект.

1) Модульные тесты позволяют исправить ошибки на ранних этапах разработки и снизить затраты.

2) Это помогает разработчикам лучше понимать кодовую базу проекта и позволяет им быстрее и проще вносить изменения в продукт.

3) Хорошие юнит-тесты служат проектной документацией.

4) Модульные тесты помогают с миграцией кода. Просто переносите код и тесты в новый проект и изменяете код, пока тесты не запустятся снова.

Протокол тестовых испытаний.

"УТВЕРЖДАЮ"

Руководитель

организации-разработчика

Пахрудинова Патимат

(подпись, фамилия)

"17" апреля 2023г.

ПРОТОКОЛ

тестовых испытаний

АИС «Учёт пациентов в регистратуре поликлиники»»

(указать наименование ПО)

"17" апреля2023г.

Настоящий протокол составлен по результатам тестирования на базе программно-технических средств МКОУ Средняя образовательная школа №4 разработчика, либо ЦОД, либо другой организации) проведенного в период с "15" апреля2023г. по "17" апреля2023г.

Условия, в которых проводилось тестирование:

Операционная система:

Windows10. Профессиональная

Система:

Процессор – Intel(R) Pentium(R) CPUG4560 @ 3.50GHz 3.50 GHz

Установленная память(ОЗУ) – 4,00 ГБ

Тип системы – 64-разрядная операционная система, процессор x64

Имя компьютера, имя домена и параметры рабочей группы:

Компьютер – pc

Тестирование проводилось по следующим режимам:

1. Компонентное/модульное тестирование
2. Интеграционное тестирование
3. Системное тестирование

Выводы по результатам тестирования:

1. Общая оценка проведения тестирования работы составила 5 (из 5)
2. Ошибки отсутствуют.

Руководитель подразделения, Пахрудинова

ответственного за проведение (фамилия, подпись)

18.04.23

Разработка алгоритма поставленной задачи и реализация его средствами автоматизированного проектирования

Алгоритм решения задачи состоит из следующих этапов:

1. Вводим данные о новом пациенте/сотруднике/назначении в предназначенные для этого поля;
2. Считываем данные, введенные в полях для ввода информации;
3. Сохраняем данные в базе данных;
4. Вносим нужные изменения;
5. Сохраняем данные в базе данных.

Входные и выходные данные будут иметь следующие типы данных:

- Фамилия Имя Отчество пациента – текстовый тип, проверка на правильность ввода;
- Лечащий врач – текстовый тип, вставляется из справочника, находящийся в таблице «Doctor»;
- Причина выписки – текстовый тип, вставляется из справочника, находящийся в таблице «Naznachenie»;
- Фамилия Имя Отчество сотрудника – текстовый тип, проверка на правильность ввода;
- Должность – текстовый тип, вставляется из таблицы «Doljnost»;
- Назначение – текстовый тип, вставляется из таблицы «Nazv»;
- Наименование – текстовый тип, проверка на правильность ввода
- ФИО врача/медсестры – текстовый тип, вводиться из таблицы «Doctor».

Выходные данные:

- Код – проставляется автоматически
- Фамилия Имя Отчество пациента – текстовый тип, проверка на правильность ввода;
- Лечащий врач – текстовый тип, вставляется из справочника, находящийся в таблице «Doctor»;
- Причина выписки – текстовый тип, вставляется из справочника, находящийся в таблице «Naznachenie»;
- Код сотрудника – проставляется автоматически
- Фамилия Имя Отчество сотрудника – текстовый тип, проверка на правильность ввода;
- Должность – текстовый тип, вставляется из таблицы «Doljnost»;
- Код пациента – вставляется автоматически в зависимости от того какому пациенту будет назначено
- Назначение – текстовый тип, вставляется из таблицы «Nazv»;
- Наименование – текстовый тип, проверка на правильность ввода
- ФИО врача/медсестры – текстовый тип, вводиться из таблицы «Doctor».

Все данные заносятся в три основные таблицы «Patcient», «Doctor» и «Naznachenie». Справочники берутся из дополнительных таблиц, о которых говорилось выше.

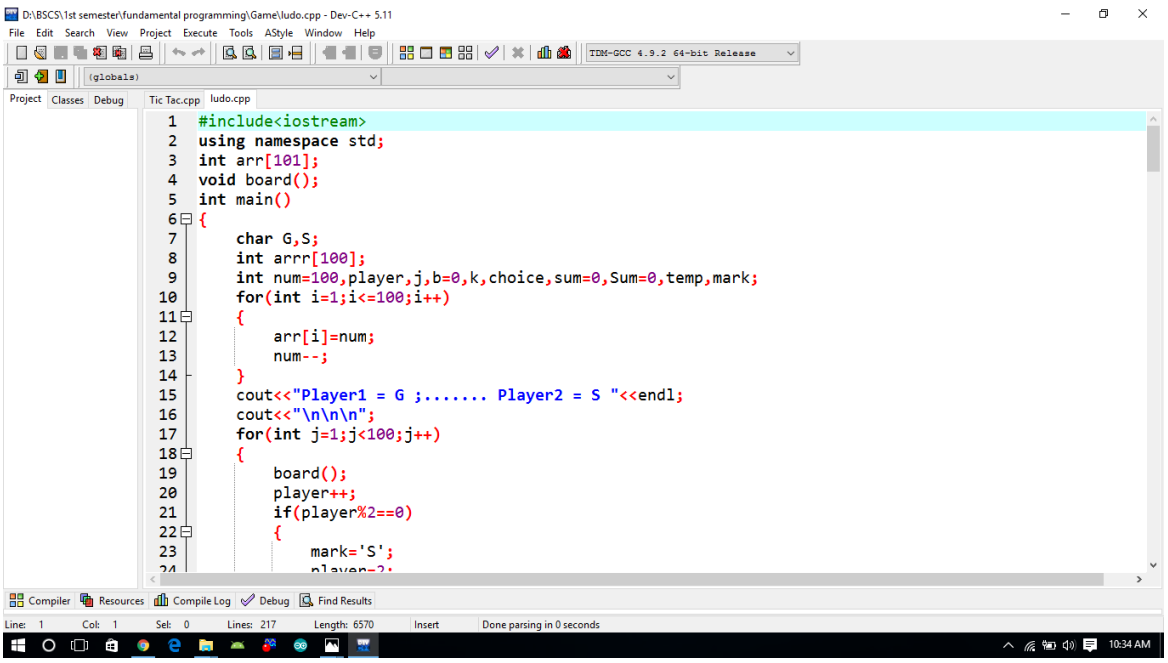


Рисунок 11 – Фрагмент из кода

19.04.23

Разработка программных модулей (в том числе приложений баз данных)

Под модульностью программ мы понимаем, прежде всего, возможность представления части исходного кода проекта, в виде бинарных независимых модулей. Более того, весь существенный код можно вынести во внешние бинарные модули, тогда как оставшийся код будет выполнять, по сути, только служебные функции.

Подобное разделение кода удобно для тестирования, отладки и быстрой компоновки отдельных частей приложения. Это может иметь смысл и для командной работы, когда каждый создает собственный модуль, а потом просто проверяет его работу путем копирования своей библиотеки в соответствующий каталог плагинов.

Однако может возникнуть ситуация, когда необходимо весь исходный код плагинов скомпилировать вместе с основным модулем. Технически это не должно вызывать проблем. Выбор должен быть на уровне параметра условной компиляции. Проще всего, просто сформировать два проекта, один для полной динамической сборки, а второй, для полной статической сборки проекта. Понятно, что допустимы и промежуточные варианты.

В нашей производственной, мы использовали оба вида компиляции. Естественно, что результат работы у них один и тот же. Только, в динамическом проекте, можно видеть работу программы вообще без плагинов

В проекте нам доступны два динамических плагина и три статических. Статические плагины присутствуют всегда, либо внутренним образом (при полной статической сборке проекта), либо в виде трех dll (расположенных рядом с exe-модулем):

- Common.dll (библиотека общего назначения, для упрощенной работы с ini-файлами, создания исходных каталогов и т.п.);
- DllLoader.dll (загрузчик динамических плагинов, который оказался достаточно сложным для включения его в библиотеку общего назначения);
- App.dll (основной код, создающий главное окно, организующий работу цикла сообщений и т.п.).

Работа этих статических плагинов (за исключением создания прототипа главного окна) на рисунке не видна, поскольку они предназначены, в основном, для обслуживания динамических плагинов.

Сами же динамические плагины представлены двумя файлами (в папке «Plugins»):

- NewWin.dll (создание множества различных дочерних MDI-окон, одного класса)
- About.dll (единственное MDI-окно, эмулирующее диалоговое окно «О программе»).

Заметим, что эти плагины имеют собственные внешние ресурсы в папке «Plugins\Res».

Если добавить плагин «NewWin.dll» в папку «Plugins», то увидим следующие изменения**.** Этот плагин создает несколько различных окон с цитатами Владимира Маяковского. При этом, первое окно центрируется, а последующие располагаются по диагонали, со смещением.

Если изменить размеры текущего окна, то следующее окно тоже изменится соответственно, только на отступы это не повлияет. Хотя это и не принципиально, но мы ограничили количество одновременно открытых окон, при превышении которых будет выдано предупреждение. Поскольку это обычное окно, а не диалог, то оно может менять фокус, размеры и местоположение. При этом размеры окна сохраняются, но, при открытии, оно всегда центрируется.

Благодаря наличию внешних ресурсов у плагинов, все эти окна полностью настраиваются. Можно менять практически все, вплоть до пиктограмм, пунктов меню, «горячих» и Alt-клавиш для них. Не говоря уже о файлах изображений и тексте в окнах.

20.04.23

Разработка тестовых наборов и тестовых сценариев

Для разработки тестовых сценариев и выполнения тестов используются системыуправления тестированием, существенно повышающие производительность тест-дизайнеров и тестировщиков, а также обеспечивающие видимость уровня качества приложений среди всех участников проекта.

Тестовые сценарии неразрывно связаны с требованиями, изменения в которых должны своевременно отражаться в тестовой документации, что позволяет сделать система управления жизненным циклом разработки приложений, при помощи механизма трассировок.

При выполнении теста тестировщик отмечает результат прохождения одного шага или всего тестового сценария, прикрепляет обнаруженные ошибки и другую вспомогательную информацию: скриншоты, дампы, логи и т.п.

Тестовые сценарии удобно объединять в тест-планы назначению:

- тестирование релиза, то есть очередной версии продукта;
- тестирование развертывания;
- тестирование удобства использования;
- конфигурационное тестирование;
- тестирование безопасности и т.п.

Зачастую ручное тестирование превращается в рутину и занимает значительное время, что отрицательно сказывается на скорости выпуска релизов. Автоматизация тестирования позволяет:

- высвободить ресурсы для проведения более сложных видов тестирования;
- снизить количество дефектов, доходящих до стадии контроля качества;
- ускорить выпуск релизов.

Сведение результатов автоматических и ручных тестов в системе управления качеством, позволяет всем участникам проекта видеть уровень

качества очередного релиза, контролировать его изменение и опираться на эту информацию при планировании своей работы.

Получение результатов тестирования и их анализ.

Получение результатов тестирования напрямую зависит от средств тестирования. В моем случае это был встроенный в 1С механизм отладки и система контроля ошибок.

Перед получением результата программа проходит несколько уровней:

- Тестирование компонентов — тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. Часто тестирование компонентов осуществляется разработчиками программного обеспечения.
- Интеграционное тестирование — тестируются интерфейсы между компонентами, подсистемами или системами. При наличии резерва времени на данной стадии тестирование ведётся итерационно, с постепенным подключением последующих подсистем.
- Системное тестирование — тестируется интегрированная система на её соответствие требованиям.
- Альфа-тестирование — имитация реальной работы с системой штатными разработчиками, либо реальная работа с системой потенциальными пользователями/заказчиком.
- Бета-тестирование — в некоторых случаях выполняется распространение предварительной версии (в случае проприетарного программного обеспечения иногда с ограничениями по функциональности или времени работы) для некоторой большей группы лиц с тем, чтобы убедиться, что продукт содержит достаточно мало ошибок. Иногда бета-тестирование выполняется для того, чтобы получить обратную связь о продукте от его будущих пользователей.

Часто для свободного и открытого программного обеспечения стадия альфа-тестирования характеризует функциональное наполнение кода, а бета-тестирования — стадию исправления ошибок.

21.04.23

Разработка проколов тестирования

Целью моего сегодняшнего дня является разработка протоколов тестирования.

Протоколы по каждому тесту должны содержать информацию, достаточную для повторения теста. Данная информация должна включать:

- план тестирования или технические требования (спецификацию) к тестированию, содержащие контрольные примеры (для каждого контрольного примера указаны его цели);
- все результаты, связанные с контрольными примерами, включая все ошибки, выявленные при выполнении теста;
- штат персонала, вовлеченного в тестирование.

Отчет о тестировании

В отчете о тестировании должны быть суммированы цели и результаты тестирования (описанные в протоколах тестирования для каждого теста). Отчет о тестировании должен иметь следующую структуру.

1. Обозначение продукта.
2. Вычислительные системы, использованные при тестировании (технические средства, программные средства и их конфигурация).
3. Использованные документы (включая их обозначения).
4. Результаты тестирования описания продукта, документации пользователя, программ и данных.
5. Перечень несоответствий требованиям.
6. Перечень несоответствий рекомендациям либо перечень не учтенных в продукте рекомендаций, либо формулировка того, что продукт не был протестирован на соответствие рекомендациям.
7. Дата окончания тестирования.

Когда продукт, который уже был протестирован, тестируется, тогда выполняются следующие требования.

22.04.23

Оформление документации на программные средства

Программная документация является неотъемлемым компонентом программного продукта и должна оформляться в соответствии с Единой системой программной документации (ЕСПД - ГОСТ серии 19). В рамках учебных работ допускается заключать всю содержательную часть программной документации в единый «отчёт по программе», при этом формальные требования к оформлению такого отчёта соответствуют требованиям к отчёту по НИР. Программная документация, кроме формальных документов (спецификация, ведомость держателей подлинников, формуляр и др.), включает:

- Техническое задание (назначение, область применения программы, требования, предъявляемые к программе).
- Текст программы (запись программы с необходимыми комментариями).
- Описание программы (сведения о логической структуре и функционировании программы).
- Пояснительная записка (схема алгоритма, общее описание алгоритма и/или функционирования программы, обоснование принятых решений).
- Эксплуатационные документы.

Программный документ «Пояснительная записка» составляется на стадии эскизного или технического проектов программы. Как правило, на стадии рабочего проекта не используется.

К эксплуатационным документам относят:

- Описание применения (сведения о назначении программы, области применения, применяемых методах, классе решаемых задач, ограничениях для применения, минимальной конфигурации технических средств).
- Руководство системного программиста (сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения).
- Руководство программиста (сведения для эксплуатации программы).

- Руководство оператора (сведения для обеспечения общения оператора с вычислительной системой в процессе выполнения программы).
- Описание языка (описание синтаксиса и семантики языка).
- Руководство по техническому обслуживанию (сведения для применения тестовых и диагностических программ при обслуживании технических средств).

Основная часть программной документации составляется на стадии рабочего проекта. Необходимость того или иного документа определяется на этапе составления технического задания. Допускается объединять отдельные виды документов. Эксплуатационный документ «Описание языка» включается в программную документацию, если разработанный программный продукт реализует некий язык программирования, управления заданиями, организации вычислительного процесса и т. П. Эксплуатационный документ «Руководство по техническому обслуживанию» включается в программную документацию, если разработанный программный продукт требует использования тестовых или диагностических программ.