ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ

**ПЕРВЫЙ МОСКОВСКИЙ ОБРАЗОВАТЕЛЬНЫЙ КОМПЛЕКС**

(ГБПОУ «1-й МОК»)

У Т В Е Р Ж Д А Ю

Председатель предметной

(цикловой) комиссии

Информационных систем

и программирования

\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Баринова Т.Г./

**ОТЧЁТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

**специальность: 09.02.07 «Информационные системы и программирование»**

по ПМ.02 «Осуществление интеграции программных моделей»

ПМ.05 «Проектирование и разработка информационных систем»

Обучающегося Зыкиной Елены Александровны

Группа 31ИС Курс 3-й 2021/2022 уч. год

Москва

2022г

**В период прохождения практики были освоены темы :**

|  |  |
| --- | --- |
|  | **ПМ.02 «Осуществление интеграции программных моделей»** |
| 12.01.2022 | Проведение предпроектных исследований. Разработка ТЗ. |
| 13.01.2022 | Составление описания на программный продукт. |
| 14.01.2022 | Создание модулей. Выбор метода разработки модуля. |
| 17.01.2022 | Программирование модуля. Логическая проверка модуля. |
| 18.01.2022 | Компиляция модуля. Отладка и тестирование модулей. |
| 19.01.2022 | Отладка модуля с целью выявления логических ошибок. |
| 20.01.2022 | Верификация и аттестация модуля. |
| 21.01.2022 | Разработка системы тестов. |
| 24.01.2022 | Выбор критерия завершенности тестирования. Тестирование ПО |
| 25.01.2022 | Апробация работы модуля. Устранение неисправностей. Зачёт. |
|  | **ПМ.05 «Проектирование и разработка информационных систем»** |
| 26.01.2022 | Анализ предметной области. |
| 27.01.2022 | Обоснование выбора технических средств. |
| 28.01.2022 | Построение и обоснование модели проекта. |
| 31.01.2022 | Проектирование и разработка интерфейса пользователя. |
| 1.02.2020 | Реализация алгоритмов. Разработка приложения. Отладка приложения. |
| 2.02.2020 | Интеграция модуля в информационную систему. |
| 3.02.2020 | Организация файлового ввода-вывода данных. |
| 4.02.2020 | Создание эмуляторов и подключение устройств. |
| 7.02.2020 | Тестирование и оптимизация мобильного приложения. |
| 8.02.2020 | Подготовка отчета. Зачет. |

Полный отчет, проект и документация по практике в электронном виде находится по адресуhttps://github.com/elena-zykina

**12.01.2022**

**Проведение предпроектных исследований. Разработка технического задания.**

Сегодня я осуществила предпроектное исследование с целью параметризации проекта создания ИС. Сначала выявляются все материальные, финансовые людские и временные ресурсы для выполнения необходимых проектных работ. На основе полученных данных я сформировала техническое задание.

Техническое задание – формируется по результатам проведённого предпроектного исследования и включает разработку и утверждение Технического задания на создание ИС.

В общем случае содержание технического задания включает следующие разделы:

1. Введение;
2. Основание для разработки;
3. Назначение разработки;
4. Технические требования к программе или программному изделию;
5. Технико-экономические показатели;
6. Стадии и этапы разработки;
7. Порядок контроля и приёмки;
8. Приложения.

**13.01.2022**

**Отладка программного обеспечения**

В этот день мы занимались отладкой разработанного программного обеспечения. А также изучили немного теории по данной теме:

Отладка — это процесс исправления ошибок, обнаруженных при тестировании программного обеспечения.

В соответствии с этапом обработки, на котором проявляются ошибки, различают:

синтаксические ошибки - ошибки, фиксируемые компилятором при выполнении синтаксического анализа программы;

ошибки компоновки - ошибки, обнаруженные компоновщиком при объединении модулей программы;

ошибки выполнения - ошибки, обнаруженные операционной системой, аппаратными средствами или пользователем при выполнении программы.

**Методы отладки программного обеспечения.**

Отладка программы в любом случае предполагает обдумывание и логическое осмысление всей имеющейся информации об ошибке. Большинство ошибок можно обнаружить по косвенным признакам посредством тщательного анализа текстов программ и результатов тестирования без получения дополнительной информации. При этом используют различные методы:

1) ручного тестирования;

2) индукции;

3) дедукции;

4) обратного прослеживания.

**16.01.2020**

**Составление описания на программный продукт**

В четверг мы составляли описание на разработанный программный продукт. Оно должно было соответствовать установленным требованиям и гостам.

Требования к структуре описания программного обеспечения по ГОСТ 34 устанавливаются [РД 50-34.698-90](http://technicaldocs.ru/%D0%B3%D0%BE%D1%81%D1%8234/%D0%BD%D0%BF%D0%B0/%D1%80%D0%B450-34.698-90). В общем случае документ должен состоять из следующих разделов:

1. Общее описание системы

Документ содержит разделы:

* назначение системы;
* описание системы;
* описание взаимосвязей АС с другими системами;

1. Проектная оценка надежности системы

Документ содержит разделы:

* введение;
* исходные данные;
* методика расчета;
* расчет показателей надежности;
* анализ результатов расчета.

1. Составление технического задания на программирование

При составлении технического задания требуется:

* определить платформу разрабатываемой программы – тип операционной системы;
* определить необходимость разработки программы, которую можно переносить на различные платформы;
* выбрать методы решения задачи;
* разработать обобщенный алгоритм решения комплекса задач, функциональную структуру алгоритма или состав объектов;
* определить требования к комплексу технических средств системы обработки информации, интерфейсу конечного пользователя;

**17.01.2020**

**Создание модулей. Выбор метода разработки модуля**

В пятницу мы изучали создание модулей и методы их разработки. После чего было необходимо разработать модуль по выбранному методу.

Современные тенденции в разработке информационных систем требуют от проектировщиков закладывать в архитектуру систем возможность динамического расширения их функционала.

**Этапы проектирования модульных приложений**

Для разработки модульного приложения, прежде всего, необходимо выделить тот функционал, который должен расширяться с помощью модулей. Далее разрабатываются интерфейсы, с помощью которых система будет обращаться к сторонним реализациям за этим функционалом. Самым тонким моментом становится вопрос о том, как динамически добавлять реализации интерфейсов.

**Структура модуля**

Очевидно, чтобы исключить лишние библиотеки из перебора, необходима дополнительная информация о модуле. Подобным источником информации может выступить текстовый файл, сопровождающий библиотеки модуля и предоставляющий информацию о них.

Простейшим решением устройства модуля может быть следующее:

1. Модуль представляет собой архив всех необходимых библиотек.

2. Кроме библиотек, модуль должен содержать дескриптор

Добавление нового модуля в систему может происходить в следующей последовательности:

1. Системе передается полный путь файла с добавляемым модулем.

2. Добавляемый модуль проверяется на соответствие своему дескриптору.

3. В директории системы создается новая поддиректория для добавляемого модуля.

4. Вычисляется уникальный идентификатор модуля.

5. Вся информация из дескриптора модуля и вычисленный идентификатор записываются в системный реестр модулей сохранения информации об использовании модуля в прошлой сессии работы в системе.

**20.01.2020**

**Программирование модуля. Логическая проверка модуля**

Модуль — это автономно компилируемая программная единица, включающая в себя различные компоненты раздела описаний и некоторые исполняемые операторы.

При разработке программного модуля целесообразно придерживаться следующего порядка:

* изучение и проверка спецификации модуля, выбор языка программирования;
* выбор алгоритма и структуры данных;
* программирование модуля;
* шлифовка текста модуля;
* проверка модуля;
* компиляция модуля.

**21.01.2020**

**Компиляция модуля. Отладка и тестирование модулей**

На сегодняшний день нашей задачей являлись компиляция модулей, а также их отладка и тестирование. Сначала мы собрали теоретическую информацию об этом процессе, а потом применили на практике.

Компиляция — сборка программы, включающая [трансляцию](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80) всех модулей программы, написанных на одном или нескольких исходных [языках программирования высокого уровня](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%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) или [языке ассемблера](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0), в эквивалентные программные модули на языке, близком [машинному коду](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4).

**Виды компиляции:**

* Пакетная.
* Построчная.
* Условная.

**22.01.2020**

**Отладка модуля с целью выявления логических ошибок**

Сегодня мы занимались отладкой модулей для выявления логических ошибок.

При разработке программ наиболее трудоемким является этап отладки и тестирования программ. Цель отладки состоит в выявлении и устранении причин ошибок.

Основными источниками ошибок являются недостаточно глубокая проработка математической модели или алгоритма решения задачи; нарушение соответствия между схемой алгоритма или записью его на алгоритмическом языке и программой, записанной на языке программирования; неверное представление исходных данных на программном бланке; невнимательность при наборе программы и исходных данных на клавиатуре устройства ввода.

В план тестирования обычно входят следующие этапы:

1. Сравнение программы со схемой алгоритма.
2. Визуальный контроль программы на экране дисплея или визуальное изучение распечатки программы и сравнение ее с оригиналом на программном бланке.
3. Трансляция программы на машинных язык.
4. Редактирование внешних связей и компоновка программы.
5. Выполнение программы.
6. Тестирование программы.

**23.01.2020**

**Верификация и аттестация модуля. Разработка системы тестов.**

В этот четверг моей задачей была разработка тестов системы. А также выполнить процесс верификации и аттестации модуля.

Верификацией и аттестацией называют процессы проверки и анализа, в ходе которых проверяется соответствие программного обеспечения своей спецификации и требованиям заказчиков. Верификация и аттестация охватывают полный жизненный цикл ПО – они начинаются на этапе анализа требований и завершаются проверкой программного кода на этапе тестирования готовой программной системы.

Верификация и аттестация не одно и то же, хотя их легко перепутать. Кратко различие между ними можно определить следующим образом:

• верификация отвечает на вопрос, правильно ли создана система;

• аттестация отвечает на вопрос, правильно ли работает система.

Согласно этим определениям, верификация проверяет соответствие ПО системной спецификации, в частности функциональным и нефункциональным требованиям. Аттестация– более общий процесс. Во время аттестации необходимо убедиться, что программный продукт соответствует ожиданиям заказчика. Аттестация проводится после верификации, для того чтобы определить, насколько система соответствует не только спецификации, но и ожиданиям заказчика.

Как уже отмечалось ранее, на ранних этапах разработки ПО очень важна аттестация системных требований. В требованиях часто встречаются ошибки и упущения; в таких случаях конечный продукт, вероятно, не будет соответствовать ожиданиям заказчика. Но, конечно, аттестация требований не может выявить все проблемы в спецификации требований. Иногда недоработки и ошибки в требованиях обнаруживаются только после завершения реализации системы.

В процессах верификации и аттестации используются две основные методики проверки и анализа систем.

1. Инспектирование ПО. Анализ и проверка различных представлений системы, например документации спецификации требований, архитектурных схем или исходного кода программ. Инспектирование выполняется на всех этапах процесса разработки программной системы. Параллельно с инспектированием может выполняться автоматический анализ исходного кода программ и соответствующих документов. Инспектирование и автоматический анализ – это статические методы верификации и аттестации, поскольку им не требуется исполняемая система.

2. Тестирование ПО. Запуск исполняемого кода с тестовыми данными и исследование выходных данных и рабочих характеристик программного продукта для проверки правильности работы системы. Тестирование – это динамический метод верификации и аттестации, так как применяется к исполняемой системе.

К методам инспектирования относятся: инспектирование программ, автоматический анализ исходного кода и формальная верификация. Но статические методы могут проверить только соответствие программ спецификации, с их помощью невозможно проверить правильность функционирования системы. Кроме того, статическими методами нельзя проверить такие нефункциональные характеристики, как производительность и надежность. Поэтому для оценивания нефункциональных характеристик проводится тестирование системы.

Несмотря на широкое применение инспектирования ПО, преобладающим методом верификации и аттестации все еще остается тестирование. Тестирование – это проверка работы программ с данными, подобными реальным, которые будут обрабатываться в процессе эксплуатации системы. Наличие в программе дефектов и несоответствий требованиям обнаруживается путем исследования выходных данных и выявления среди них аномальных. Тестирование выполняется на этапе реализации системы.

На разных этапах процесса разработки ПО применяют различные виды тестирования.

1. Тестирование дефектов проводится для обнаружения несоответствий между программой и ее спецификацией, которые обусловлены ошибками или дефектами в программах. Такие тесты разрабатываются для выявления ошибок в системе, а не для имитации ее работы.

2. Статистическое тестирование оценивает производительность и надежность программ, а также работу системы в различных режимах эксплуатации. Тесты разрабатываются так, чтобы имитировать реальную работу системы с реальными входными данными. Надежность функционирования системы оценивается по количеству сбоев, отмеченных в работе программ. Производительность оценивается по результатам измерения полного времени выполнения операций и времени отклика системы при обработке тестовых данных.

Главная цель верификации и аттестации – удостовериться в том, что система "соответствует своему назначению". Соответствие программной системы своему назначению отнюдь не предполагает, что в ней совершенно не должно быть ошибок. Скорее, система должна достаточно хорошо соответствовать тем целям, для которых планировалась. Уровень необходимой достоверности соответствия зависит от назначения системы, ожиданий пользователей и условий на рынке программных продуктов.

**Планирование верификации и аттестации**

Верификация и аттестация – дорогостоящий процесс. Для больших систем, например систем реального времени со сложными нефункциональными ограничениями, половина бюджета, выделенного на разработку системы, тратится на процесс верификации и аттестации. Поэтому очевидна необходимость тщательного планирования данного процесса.

В процессе планирования верификации и аттестации необходимо определить соотношение между статическими и динамическими методами проверки системы, определить стандарты и процедуры инспектирования и тестирования ПО, утвердить технологическую карту проверок программ и составить план тестирования программ. Чему уделить больше внимания – инспектированию или тестированию, зависит от типа разрабатываемой системы и опыта организации. Чем более критична система, тем больше внимания необходимо уделить статическим методам верификации.

**24.01.2020**

**Выбор критерия завершенности тестирования. Апробация работы модуля**

Сегодня мы продолжили тестирование системы и привели его к завершению, обозначив критерии завершенности. А также изучили понятие апробации и применили знания на практике.

**Формы апробации:**

Апробация может проходить в форме прилюдных отчетов, дискуссий, дискуссий, и еще в форме рецензирования представленных работ. Официальная апробация исполненных работ нередко связана с их общественной охраной. Присутствует и играет не самую последнюю роль и неофициальная апробация: разговоры, споры с экспертами и коллегами. Апробацию диссертации надо начинать сразу после начала работы над ней, в данном случае, кандидат сумеет получить беспристрастную оценку каждого шага проделанного им исследования, произведенных выводов и практических советов, что в нем содержатся. Проинформировать научную общественность о материалах собственной диссертации и работе над ней кандидат может, принимая участие в разных конференциях и симпозиумах, выступая на заседаниях кафедры, участвуя в разных событиях, проводимых научным обществом, кроме того у него есть возможность готовить и подталкивать во всевозможные органы собственные предложения по исследуемой теме. Помимо того, кандидат может публиковать книги, тезисы отчетов на конференциях, депонированные части собственных научных изысканий. Работу надо каждый день оговаривать с сотрудниками, преподавательским составом и научными работниками того учебного заведения, в каком готовится диссертация.

**Значение апробации:**

Есть основания признать, что могут быть полезны в конечном счете всевозможные образовавшиеся в ходе представления работы вопросы. Все связано с тем, как они воспринимаются и применяются. Из числа вопросов можно выделить уточняющие, связанные с недопониманием, неполным или же неверным осознанием изложенного. Они побуждают к уточнению, поиску более точных формулировок, доработке стиля, т.е. могут помочь сделать изложение более точным и убедительным. Вопросы, дополняющие предполагают запрос на вспомогательную информацию об источниках и перспективах становления, прецедентах, первопричинах, следствиях и т.д. Они побуждают исследователя использовать новейшие прецеденты, отдать вспомогательные оценки и мониторинги. Такие вопросы расширяют спектр оценок и раскладов. Вопросы, корректирующие побуждают к уточнению, усиливают аргументацию, уничтожают двусмысленность. Проблемные вопросы вскрывают трудности, нацеливают на более основательную интерпретацию, выводят на свежие проблемы и задачи. Стоит посоветовать чрезвычайно тщательно и дружественно воспринимать все оценки, рекомендации, советы и соратников, и оппонентов, в том числе и критические.

**27.01.2020**

**Предметная область**

Сегодня на практике я занималась анализом предметной области.

Предметную область можно определить как сферу человеческой деятельности, выделенную и описанную согласно установленным критериям. В описываемое понятие должны входить сведения об ее элементах, явлениях, отношениях и процессах, отражающих различные аспекты этой деятельности. В описании предметной области должны присутствовать характеристики возможных воздействий окружающей среды на элементы и явления предметной области, а также обратные воздействия этих элементов и явлений на среду. Работа по изучению и анализу предметной области: проектировании интеллектуальных систем оказывает решающее влияние на эффективность ее работы.

Специфика предметной области может оказывать существенное влияние на характер функционирования проектируемой интеллектуальной системы, выбор метода представления знаний, способов рассуждения о знаниях, и т. д.

Предметную область можно определить как объект или производственную систему со всем комплексом понятий и знаний о ее функционировании. При исследовании проблемной области необходимы знания о задачах, решаемых в производственной системе, и стоящих перед ней целях. Определяются также возможные стратегии управления и эвристические знания, используемые в процессе эксплуатации производственной системы.

**Анализ предметной области**

Одна из первых задач, с решением которых сталкивается разработчик программной системы — это изучение, осмысление и анализ предметной области. Дело в том, что предметная область сильно влияет на все аспекты проекта: требования к системе, взаимодействие с пользователем, модель хранения данных, реализацию и т.д.

Функциональная область действия определяет услуги, предоставляемые системой, и вначале до конца неизвестны. При его разработке в первую графу вписываются имена основных действующих лиц, т.е. тех, кто имеет цели, во вторую графу - цель каждого действующего лица, а в третью - приоритет или предположение о том, в какую версию войдет эта услуга.

**28.01.2020**

**Обоснование выбора технических средств**

В начале третьей недели мы занимались обоснование выбора технических средств, научившись делать это правильно.

На основании поставленной задачи делается вывод о необходимости использования той или иной программы. После чего выбирается выбор технического средства (ПК).

1. Технические характеристики: -объем памяти; -быстродействие; -состав аппаратных средств;
2. Удобство и простота использования;
3. Стоимостные затраты;

При выборе комплекса технических средств необходимо учитывать следующие факторы:

1. стоимость приобретения технических средств;
2. стоимость использования технических средств;
3. надежность технических средств и срок службы;
4. удобство комплекса технических средств (КТС);

**29.01.2020**

**Построение и обоснование модели проекта**

Так же на этой недели нашей задачей являлось построение и обоснование модели проекта. Этим мы и занимались сегодня.

Информационное обеспечение является базой, на которой строится вся управленческая деятельность. Информацию здесь следует рассматривать как некую совокупность различных сообщений, сведений, данных о соответствующих предметах, явлениях, процессах, отношениях и т.д. Эти сведения, будучи собранными, систематизированными и преобразованными в пригодную для использования форму играют в управлении исключительную роль. Необходимым условием для успешного функционирования любого промышленных предприятий является нормальная работа следующих процессов: - целенаправленный сбор, первичная обработка информации; - организация каналов доступа пользователей к собранной информации; - своевременное использование собранной информации для принятия решений. Основная проблема сбора необходимой информации состоит в том, чтобы обеспечить: - полноту, адекватность, непротиворечивость и целостность информации; - минимизацию технологического запаздывания между моментом зарождения информации и тем моментом, когда к информации может начаться доступ. Обеспечить это можно только современными автоматизированными методиками, базирующимися на основе информационных систем. Крайне важно, чтобы собранная информация была структурирована с учетом потребностей потенциальных пользователей и хранилась в форме, позволяющей использовать современные технологии доступа и обработки. Разработка информационных систем управления (ИСУ) является довольно сложным процессом, который требует значительного времени и ресурсов. Современные крупные проекты ИСУ характеризуются, как правило, следующими особенностями: - сложность описания (достаточно большое количество функций, процессов, элементов данных и сложные взаимосвязи между ними), требующая тщательного моделирования и анализа данных и процессов; - наличие совокупности тесно взаимодействующих компонентов (подсистем), имеющих свои локальные задачи и цели функционирования (например, традиционных приложений, связанных с обработкой транзакций и решением регламентных задач, и приложений аналитической обработки (поддержки принятия решений), использующих нерегламентированные запросы к данным большого объема); - отсутствие прямых аналогов, ограничивающее возможность использования каких-либо типовых проектных решений и прикладных систем; - разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и сложившимся традициям использования тех или иных инструментальных средств; - существенная временная протяженность проекта, обусловленная, с одной стороны, ограниченными возможностями коллектива разработчиков, и, с другой стороны, масштабами организации-заказчика и различной степенью готовности отдельных ее подразделений к внедрению ИСУ. Для успешной реализации проекта объект проектирования (ИСУ) должен быть прежде всего адекватно описан, должны быть построены полные и функциональные непротиворечивые и информационные модели ИСУ. Кроме того, в процессе создания и функционирования ИСУ информационные потребности пользователей могут изменяться или уточняться, что усложняет разработку и сопровождение таких систем. В настоящее время один из наиболее сложных и важных этапов разработки ИСУ, этап построение информационной модели, остается во многом не формализованным. Начальные фазы проекта имеют решающее влияние на достигаемый результат, так как в них принимаются основные решения, определяющие качество информационной системы. Доля вклада в конечный результат концептуальной фазы достигает 30%.

В результате проведения первичной обработки получаем не массу несистематизированной первичной информации, а информационную базу, содержащую данные о предприятии, пригодную для дальнейшего непосредственного автоматического анализа. Именно такое представление результатов обследования мы будем использовать в дальнейшей работе. Поставим задачу построения информационной модели, которая позволяла бы отражать не только связи между структурными подразделениями предприятия и их вес, но и оценивать суть происходящих в организации процессов. В данном случае – какие операции (функции) над информацией (документами) выполняются внутри организации. В дальнейшем будем называть такую информационную модель функциональной ориентированной. Получение такой модели позволит ставить задачу оптимизации организационной структуры предприятия по новым критериям, например, по загруженности отдельных функций, по равномерной загрузки и т.д. Полученная функционально-ориентированная модель позволит реализовать методику построения организационной модели предприятия «снизу»: на первом этапе выявить весь перечень функций, которые должны быть реализованы на данном предприятии, для эффективного управления и достижения поставленной цели; определить внутренние и внешние связи между функциями; оценить количество информации проходящей по этим связям; провести реорганизацию отделов и служб путем перераспределения данных функций по признаку подобия функций.

В настоящее время наибольшее распространение получили модели для следующих классов задач: - составление расписаний и календарное планирование; - массовое обслуживание; - распределение; - управление запасами; - износ и замена оборудования; - конфликтные ситуации. В поддержку создания служб и систем управления потоками работ предлагаются методологии, стандарты и специализированное программное обеспечение, образующие в комплексе инструментальные средства разработчика. Методология – это совокупность методов, применяемых в жизненном цикле разработки процесса и объединенных одним общим подходом. В настоящее время получила распространение методология SADT. (Structured Analysis and Design Tecchnique) ─ методология структурного анализа и проектирования, которые дают ряд преимуществ в системах управления: - формализацию описания потоков работ; - переносимость: модели процессов, созданные в рамках одной системы, могут работать под управлением другой системы; - универсальность: применение единого механизма описания управления потоками работ в различных сферах деятельности. На данный момент разработан ряд стандартов для описания реальных потоков работ, которые можно разбить на две категории: - Графовые модели, отражающие древовидную структуру процесса. - Блочные модели, наиболее приближенные к блочной структуре языков программирования. Итак, модели, указанные выше, в ряде случаев удобно использовать для эффективного описания окружения системы и ее функционирования. Между тем, по большому счету определяющей в любой системе является информация о потоках, обслуживающих систему. Поэтому имеет смысл исследовать не просто модель системы, а ее информационную модель (ИМ) детализированную до уровня функций и ее заполняемость информационными потоками, а не блоков системы. Такую модель можно называть функционально ориентированной ИМ (ФИМ). Использование ФИМ важно и для построения модели функциональной структуры системы. Использование ФИМ позволяет ставить и решать новые задачи на уровне организационной и функциональной структуры, например, определять загрузку функций, перераспределять (оптимизировать) документооборот между отдельными функциями с целью обеспечения их полной загрузки и т.д. Прежде, чем обсуждать эффективность ФИМ надо обратить внимание на то, что само исходное понятие информации оказывается до сих пор неоднозначно.

**30.01.2020**

**Проектирование и разработка интерфейса пользователя**

Сегодня нашей задачей была разработка пользовательского интерфейса и изучение материала по этой теме.

Что такое UI? Интерфейс — общая граница между двумя функциональными объектами, требования к которой определяются стандартом; совокупность средств, методов и правил взаимодействия (управления, контроля и т. д.) между элементами системы (источник: wikipedia.org). Это точное, но скучноватое определение. Пользовательский интерфейс (UI) — это «способ, которым вы выполняете какую-либо задачу с помощью какого-либо продукта, а именно совершаемые вами действия и то, что вы получаете в ответ». В повседневной жизни мы постоянно сталкиваемся с интерфейсами. Это и сайты соцсетей, и элементы управления в салоне автомобиля, и пульт ДУ для телевизора, и голосовое управление умным домом, и панель кнопок в лифте. Выходит, мы используем один продукт для управления другим продуктом. Но давайте не будем перечислять все явления в нашей жизни, а поговорим непосредственно о веб-сервисах и приложениях и о том, как сделать их использование удобным. Современные веб-сервисы и системы в корне меняют работу людей. Программный интерфейс не только решает нашу проблему взаимодействия с приложением, но и делает это взаимодействие максимально комфортным. Нам важно наличие интерфейса, позволяющего при меньшем количестве усилий ознакомиться с возможностями приложения и понять принципы работы в нём.

Под интерфейсом понимается любой экранный информационный или интерактивный интерфейс. Таковыми являются:

* сайты,
* мобильные приложения,
* приложения для стационарных компьютеров,
* презентационные панели,
* информационные стационарные экраны.

Проецируемая картинка на стену или полотно с использованием проектора и управляемая жестами или голосом тоже считается интерфейсом.

Этапы разработки

Полный цикл разработки интерфейса включает следующие этапы:

* Исследование
* Пользовательские сценарии
* Структура интерфейса
* Прототипирование интерфейса
* Определение стилистики
* Дизайн концепция
* Оформление всех экранов
* Анимация интерфейса
* Подготовка материалов для разработчиков

**31.01.2020**

**Технология разработки программного обеспечения**

Сегодня на практике мы занимались разработкой алгоритмов и отладкой приложений.

Процесс разработки программного обеспечения — структура, согласно которой построена разработка программного обеспечения. Существует несколько моделей такого процесса, каждая из которых описывает свой подход, в виде задач и/или деятельности, которые имеют место в ходе процесса. Шаги процесса

Процесс разработки состоит из множества подпроцессов, или дисциплин, некоторые из которых показаны ниже. В модели водопада они идут одна за другой, в других аналогичных процессах их порядок или состав изменяется.

* Анализ требований;
* Спецификация программного обеспечения;
* Проектирование программного обеспечения;
* Программирование;
* Тестирование программного обеспечения;
* Системная интеграция;
* Внедрение программного обеспечения;
* Сопровождение программного обеспечения.

**Тестирование и отладка программного обеспечения**

Цели, с которыми производится тестирование программного обеспечения вполне очевидны. Первая задача, которая стоит перед командой испытателей – выполнить наглядную демонстрацию того, что продукт действительно работает. Это необходимо как самим разработчикам, так и заказчикам проекта. От вердикта последних зависит попадет программа к конечному потребителю, или нет. Вторая цель, реализации которой способствует тестирование – определение возможных изъянов разработанного продукта. Речь идет, в первую очередь, о ситуациях, в которых программное обеспечение может повести себя ненадлежащим образом. В ходе тестирования программного обеспечения, как и при любых других способах проверки продукции, определяется уровень качества ПО. Испытатели должны убедиться в том, что объект соответствует всем, предъявляемым к нему требованиям. Последние не очень многочисленны, но при этом, чрезвычайно конкретны. В первую очередь, оценивается функциональная пригодность продукта. О ней свидетельствует тот факт, что программа выполняет все функции, которые от нее требуются. Вторая характеристика, которая подлежит проверке – надежность. Тестирование позволяет выяснить, сможет ли программа выполнять все, возложенные на нее функции в заданных условиях. Третий фактор, который необходимо выяснить в ходе проверки – уровень производительности продукта. Здесь учитывается коэффициент полезного действия объекта исследования.

**03.02.2020**

**Интеграция модуля в информационную систему**

В этот день я занималась интеграцией разработанного модуля в информационную систему.

Интеграция модуля в информационную систему:

* Уровень данных

Создание дополнительного протокола или базы данных для удобного обмена информацией между приложениями, которые имеют различные форматы и типы данных.

* Программные и пользовательские интерфейсы

Разработка унифицированного интерфейса, реализующего прикладную бизнес-логику и позволяющего эффективно взаимодействовать со всеми компонентами, сохраняя неизменными слои обработки данных.

* Функционально-прикладной и организационный уровень

Консолидация однотипных функций и алгоритмов путем перераспределения потоков данных, перестройки организационных структур, а также модификации схемы информационного взаимодействия.

* Корпоративные программные приложения

Интеграция приложений осуществляется посредством включения дополнительных модулей, способных обращаться к нескольким системам, в том числе разными способами.

* Web-сервисы

Обеспечение доступа ко всем компонентам системы с использованием стандартного web-интерфейса.

Интеграция позволяет оптимизировать информационную систему за счет объединения приложений, используемых в рамках общего бизнес-процесса, но не связанных технологически.

В результате интеграции формируется цельная унифицированная экосистема приложений, позволяющая повысить эффективность работы, обеспечить удобство доступа к различным системам и значительно снизить затраты на внедрение новых программных решений и сервисов.

**04.02.2020**

**Файловый и потоковый ввод-вывод**

Сегодня на практике мы занимались организацией файлового ввода-вывода данных. Для этого нам потребовались следующие теоретические материалы:

Файловый и потоковый ввод-вывод относятся к передаче данных с носителя информации или на него. В .NET Framework пространства имен System.IO содержат типы, которые обеспечивают как синхронные, так и асинхронные операции чтения и записи для потоков и файлов. Кроме того, эти пространства имен содержат типы, выполняющие сжатие и распаковку файлов, а также типы, которые обеспечивают взаимодействие через каналы и последовательные порты.

При работе с файлами используются пути к каталогам, запоминающие устройства, а также имена файлов и каталогов. В отличие от файла, поток — это последовательность байтов, которую можно использовать для записи или чтения из вспомогательного запоминающего устройства, являющегося одним из устройств хранения информации.

**Файлы и каталоги**

Типы в пространстве имен System.IO можно использовать для взаимодействия с файлами и каталогами. Например, можно получать и задавать свойства файлов и каталогов, а также извлекать коллекции файлов и каталогов на основе критерия поиска.

Соглашения об именовании и способы указать путь к файлу в системах Windows, включая синтаксис устройств DOS, поддерживаемый в .NET Core 1.1 и более поздних версиях и платформе .NET Framework 4.6.2 и более поздних версиях.

**Асинхронные операции ввода-вывода**

Чтение и запись больших объемов данных может быть ресурсоемкой. Эти задачи необходимо выполнять асинхронно, если приложение должно продолжать отвечать на запросы пользователя. В случае синхронных операций ввода-вывода поток пользовательского интерфейса будет заблокирован до тех пор, пока ресурсоемкая операция не завершится. При разработке приложений Microsoft Store для Windows 8.x используйте асинхронные операции ввода-вывода, чтобы не создавалось впечатления, что приложение прекратило свою работу.

**Операции ввода-вывода в приложениях Microsoft Store**

.NET для приложений Магазина Windows 8.x содержит множество типов для чтения и записи в потоки, однако этот набор содержит не все типы ввода-вывода платформы .NET Framework.

Следует отметить некоторые важные различия в использовании операций ввода-вывода в приложениях Microsoft Store для Windows 8.x:

* Специальные типы, относящиеся к операциям с файлами, такие как File, FileInfo, Directory и DirectoryInfo, не включены в .NET для приложений Магазина Windows 8.x. Вместо этого используйте типы в пространстве имен Windows.Storage среды выполнения Windows, например StorageFile и StorageFolder.
* Изолированное хранилище недоступно; вместо этого используйте [данные приложения](https://docs.microsoft.com/previous-versions/windows/apps/hh464917(v=win.10)).
* Используйте асинхронные методы, такие как ReadAsync и WriteAsync, чтобы предотвратить блокировку потока пользовательского интерфейса.

**05.02.2020**

**Создание эмуляторов и подключение устройств**

Сегодня на практике мы создавали эмуляторы и подключали устройства. И нам объясняли следующую информацию:

**Эмуляторы**

Тестирование на физических устройствах – штука оправданная. Но это не значит, что не нужно тестировать на эмуляторах. Они позволяют расширить тестируемый набор устройств и проверять изменения на лету.

##### **Эмулятор Android**

Для начала, скачайте подборку, включающую Android Development Tools (ADT) для Eclipse и Android software development kit (SDK). Затем следуйте инструкциям по установке, не забудьте кроме настроек по-умолчанию установить и “Intel x86 Emulator Accelerator (HAXM installer)”. Также вам понадобится установить HAXM (IntelHaxm.dmg на Маке и IntelHaxm.exe на PC)

Затем создайте Android virtual device (AVD) для тестируемого устройства. В менеджере AVD есть список готовых устройств в “DeviceDefinitions”. Для начала выберите одно из них и нажмите “Create AVD”

Выберите любой CPU, и поставьте “No skin“ и “Use host GPU”. Теперь можно запускать виртуальное устройство и использовать браузер Android для тестирования.

**Подключение устройств**

**Android**

В случае с Андроидом инструменты удалённой отладки позволяют работать с десктопа при помощи Chrome’s Developer Tools. Кроме того, инструменты эти опять-таки кросс-платформенные.

##### Сначала, на телефоне пройдите в “Settings” → “About Phone”, или “Settings” → “About Tablet”. Затем нужно нажать на “Build Number” семь раз. Кроме шуток. После этого вы увидите сообщение насчёт разработчика.

##### Затем обратно в основных настройках “Developer Options” нужен пункт “USB debugging”.

В адресной строке браузера Chrome наберите about:inspect. Разрешите “Discover USB devices”, и вы увидите в меню своё МУ.

**06.02.2020**

**Тестирование и оптимизация мобильного приложения**

К концу практики мы занимались тестированием мобильного приложения, а также его оптимизацией.

**Тестирование требований**

Тестировщик анализирует требования на полноту и противоречивость. В каждом проекте исходные требования содержат противоречивую информацию. Мы их решаем еще до начала разработки. Так же в каждом проекте требования неполны: не хватает макетов второстепенных экранов, ограничений на поля ввода, отображения ошибок, кнопки никуда не ведут. Неочевидны невидимые на макетах вещи: анимации, кеширование картинок и содержимого экранов, работа в нестандартных ситуациях.

Если менеджер проекта поставит галочку «для тестирования», тестировщикам уходит письмо о новой сборке для тестирования.

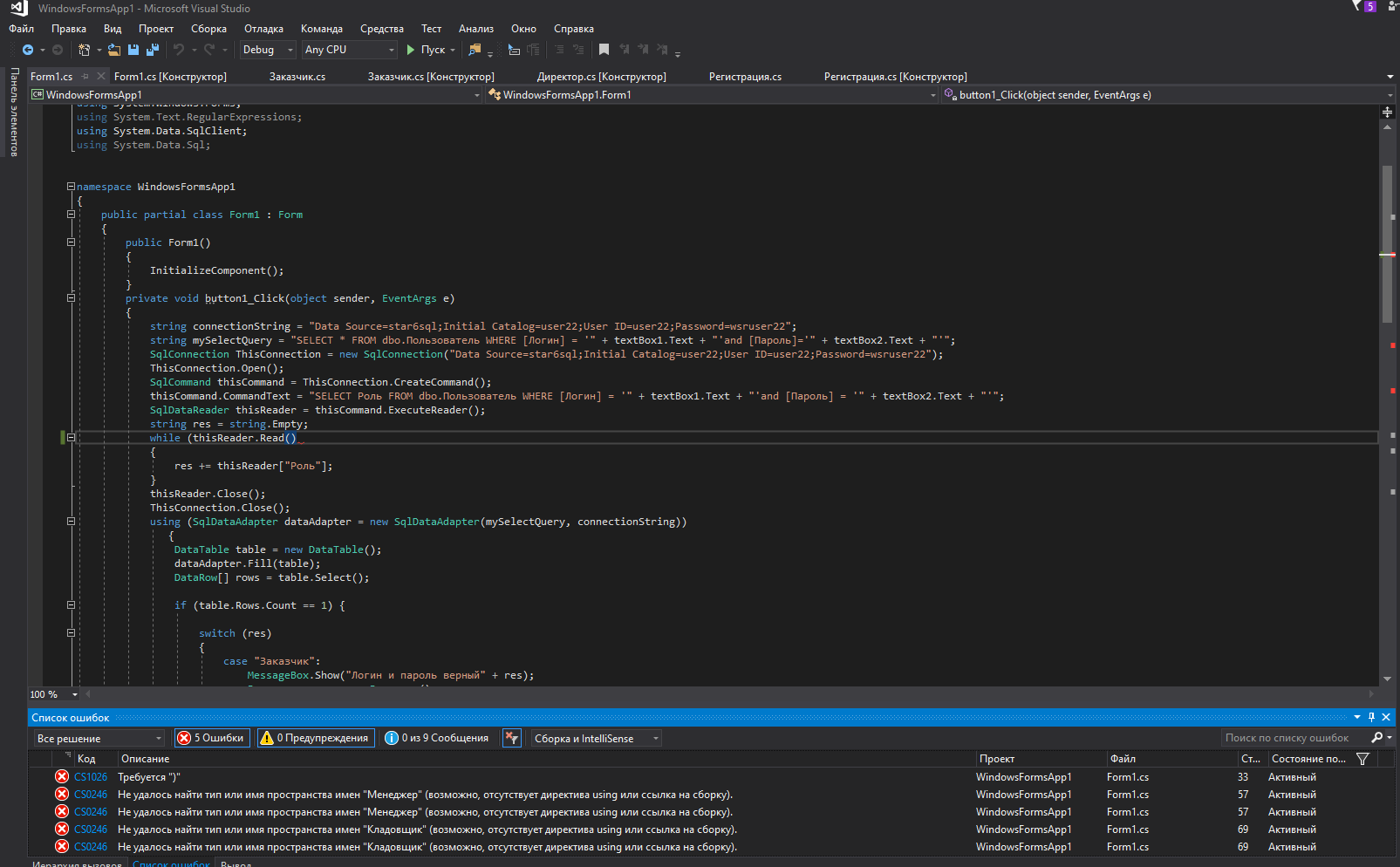
**07.02.2020**

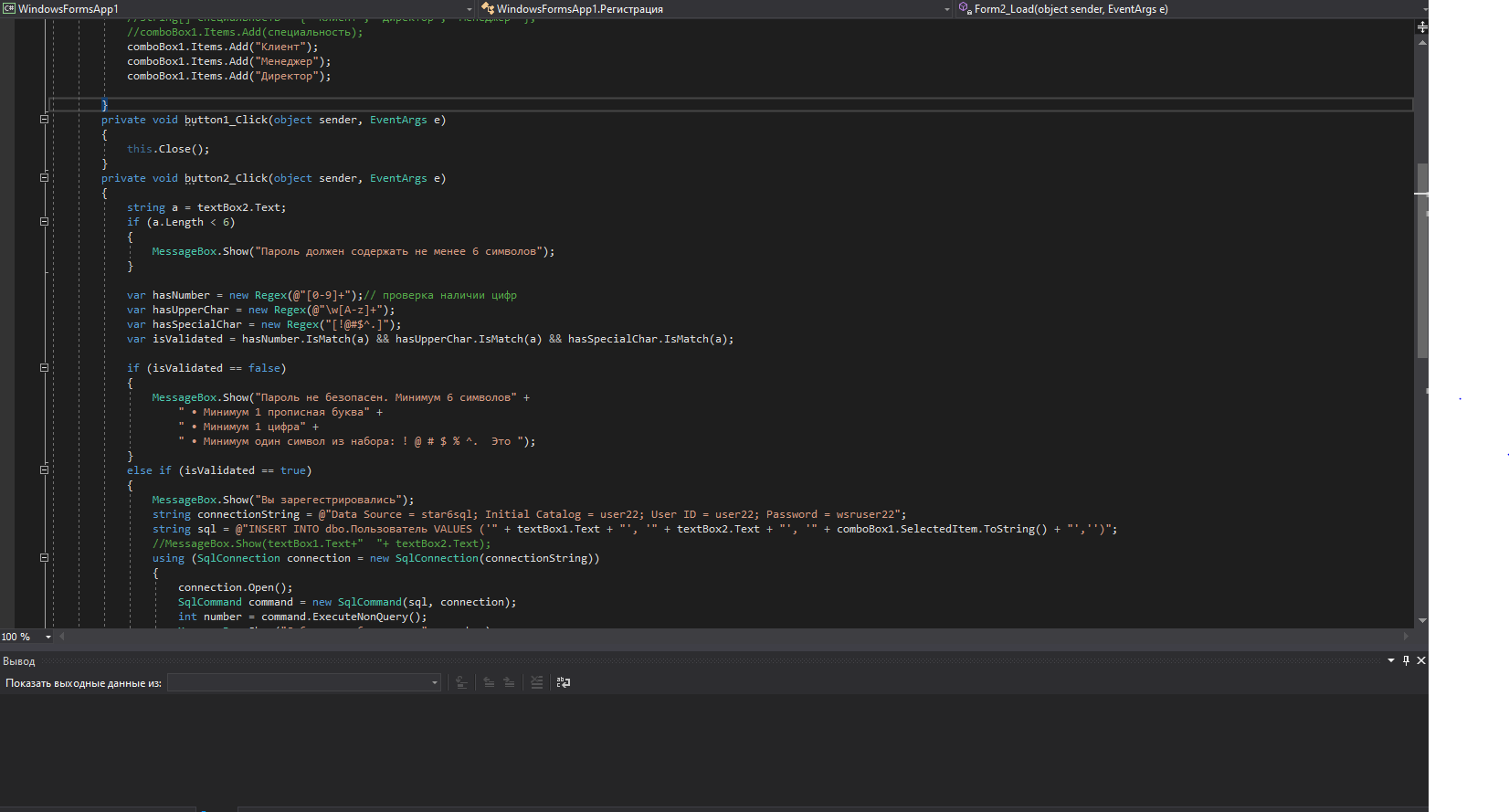
**Подготовка отчета. Зачет.**

В последний день практики нашей задачей являлось создание отчета по выполненной работе и защита учебной практики.

За время прохождения учебной практики я получала новые знания и совершенствовала практические навыки. Каждый день я фиксировала в дневник практики освоенный за день материал, чтобы закрепить свои знания, а также использовать их в дальнейшей работе по специальности. Самое главное, понимание выданной темы и поиск информации по ней. Объем информации в различных источниках разнообразен и очень велик, его нужно уметь отфильтровать и качественно обработать. В конце учебной практики необходимо было соединить весь собранный материал в один отчет, заполнить дневник практики недостающими данными, распечатать все документы. Также к концу практики от меня требовалось создать презентацию для отчета по темам, которые изучались в рамках учебной практики, на слайдах должны находиться темы и соответствующие графические изображения. Отчет о работе необходимо представить в виде защиты практики.

По результатам задания №3 проведен поиск ошибок и отладка программы с использованием специализированных программных средств Visual Studio.

До отладки программы:

После отладки программы:

В результате выполнения задания №4 был разработан графический интерфейс приложения «Цветочек»:

