**Квал. экзамен. Теория.**

**Билеты 1, 11, 21.**

**Методы и технологии программирования. Обосновать выбор методов и технологий разработки ПО, реализованного в ходе курсового проектирования.**

**Технологией программирования** называют совокупность методов и средств, используемых в процессе разработки программного обеспечения. Как любая другая технология, технология программирования представляет собой набор технологических инструкций, включающих:

− указание последовательности выполнения технологических операций;

− перечисление условий, при которых выполняется та или иная операция;

− описания самих операций, где для каждой операции определены исходные данные, результаты, а также инструкции, нормативы, стандарты, критерии и методы оценки и т. п.

Кроме набора операций и их последовательности, технология также определяет способ описания проектируемой системы, точнее модели, используемой на конкретном этапе разработки. (модели жизненного цикла).

МЕТОДОЛОГИИ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ:

**Структурное программирование**.

Принципы:

1. запрет на использование оператора GOTO.

2. любая программа строится из трёх базовых управляющих конструкций: последовательность, ветвление, цикл.

3. в программе базовые управляющие конструкции могут быть вложены друг в друга произвольным образом.

4. повторяющиеся фрагменты программы можно оформить в виде подпрограмм (процедур и функций).

**Визуальное программирование** — способ создания программы для ЭВМ путём манипулирования графическими объектами вместо написания её текста. Визуальное программирование часто представляют как следующий этап развития текстовых языков программирования. Визуальное программирование (результатом визуального проектирования является заготовка будущей программы, в которую уже внесены соответствующие коды).

**Объектно-ориентированное программирование (ООП)** — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования

**Класс** – множество объектов, связанных общностью структуры и поведения (класс содержит описание структуры и поведение всех объектов, связанных отношением общности).

В языках программирования понятие объекта реализовано как совокупность свойств (структур данных, характерных для данного объекта), методов их обработки (подпрограмм изменения их свойств) и событий, на которые данный объект может реагировать.

**?Облачные технологии** – это удобная среда для хранения и обработки информации, объединяющая в себе аппаратные средства, лицензионное программное обеспечение, каналы связи, а также техническую поддержку пользователей.

**Мо́дульное программи́рование** — это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам.

**Билет 2, 12, 22.**

**Понятие спецификации программного продукта. Обосновать основные спецификации программного модуля, реализованного в ходе курсового проектирования.**

**Спецификация программного продукта** — Это документ, который содержит полное и четкое описание разрабатываемого продукта. Так как спецификация служит в том числе и для получения обратной связи от клиента, написана она должна быть в простой и легкой для восприятия форме.

**Основные свойства:**

1. **Спецификация не должна содержать деталей реализации**. В отличие от программы она "говорит", что надо сделать, а не как это делать.

2. **Спецификация должна обладать формальностью** (однозначностью прочтения, точностью), причем диапазон требований здесь очень широк: от полностью формализованного описания до слегка формализованного. Описание на "естественном языке" обычно считается неудовлетворительным, поскольку оно слишком неформально.

3. **Спецификация должна быть понятной** (ясной, читабельной). В этом заключается еще одно отличие от программы. В общем случае спецификация должна быть более понятным описанием задачи, чем программа, так как краткость не всегда содействует ясности и понятности.

4. **Спецификация должна обладать полнотой описания задачи**: ничто существенное не должно быть упущено.

**Основные положения спецификации:**

1. **Предварительные исследования**. Оценивается степень удовлетворенности пользователей существующими программными продуктами и аппаратными средствами. Этот этап должен быть по возможности коротким и дешевым.

2. **Формирование и анализ требований**. Формируются системные требования путем изучения существующих аналогичных систем. Этот этап может включать разработку нескольких моделей системы и ее прототипов, что помогает сформировать функциональные требования к системе.

3. **Специфицирование требований**. Осуществляется перевод всей совокупности информации, собранной на предыдущем этапе, в документ, определяющий множество требований. Этот документ обычно содержит два типа требований: пользовательские – обобщенные представления заказчиков; системные – детальное описание функциональных показателей системы.

4. **Утверждение требований**. Проверяется выполнимость, согласованность и полнота множества требований. В процессе формирования ограничений неизбежно возникновение каких-либо ошибок. На этом этапе они должны быть по возможности выявлены и устранены.

**Для чего нужна Спецификация:**

1. Можно получить точную оценку стоимости, рисков и затрат времени

2. Клиент сможет более четко сформировать собственное видение проекта

3. Заказчик и Исполнитель будут иметь одинаковое представление о продукте

4. Она поможет выявить оптимальный набор функций

5. Она служит основой для формирования другой технической документации

6. Процесс разработки будет оптимизирован — минимизированы затраты времени

7. Никакого дублирования задач

8. Позволяет структурировать проблемы, чтобы решать их проще и быстрей

9. Она помогает понять, какие результаты считаются оптимальными при тестировании

**Билет 3, 13, 23.**

**Порядок оптимизации программного продукта. Подготовить план оптимизации программного кода модуля, реализованного в ходе курсового проектирования.**

Оптимизация стоит на трех «китах» — естественность, производительность, затраченное время.

1. **Естественность**. Код должен быть аккуратным, модульным и легко читабельным. Каждый модуль должен естественно встраиваться в программу. Код должен легко поддаваться редактированию, интегрированию или удалению отдельных функций или возможности без необходимости вносить серьезные изменения в другие части программы.

2. **Производительность**. В результате оптимизации вы должны получить прирост производительности программного продукта. Как правило, удачно оптимизированная программа увеличивает быстродействие минимум на 20-30% в сравнение с исходным вариантом.

3. **Время**. Оптимизация и последующая отладка должны занимать небольшой период времени. Оптимальными считаются сроки, не превышающие 10 – 15 % времени, затраченного на написание самого программного продукта. Иначе это будет нерентабельно.

**Первое, что необходимо сделать, это выявить «узкие места» программы.** Нет смысла трогать тот кусок программы, где и без вас все работает прекрасно. Здесь вы вряд ли что-то выиграете при оптимизации. В первую очередь, стоит обратить внимание на блоки кода, которые регулярно или часто повторяются в процессе работы – циклы и подпрограммы.

**Не стоит трогать единичные операнды**, поскольку работают они крайне редко и толку в их модификации нет никакого. Они отработают один раз, и больше к этому коду обращений не будет.

**Также вы не сумеете добиться достойных результатов в случае обращения к внешним устройствам и другим программным системам**. До и после таких фрагментов можно что-то ускорить. Но там, где задержка может возникать по причине взаимодействия с внешними данными, лучше предусмотрите заглушку типа «Подождите, операция может занять несколько минут».

**Оптимизация кода не слишком отличается от обычного исправления багов**. Более того, с их устранения и начинается работа по оптимизации программы.

**Первым делом нужно проверить код на наличие устаревших или вообще ненужных фрагментов**. Таких исполняемых модулей или веток в большой программе находится обычно много. Что-то написали, но оказалось, что функционал не нужен, и его просто забыли удалить. Другие части оказались не нужны в результате очередного обновления. Все они занимают место. А некоторые продолжают исполняться, хоть в этом нет никакого смысла. И, таким образом, замедляют работу системы.

**Билет 4, 14, 24.**

**Порядок и технологии тестирования программных продуктов. Подготовить акт тестирования программного кода модуля, реализованного в ходе курсового проектирования.**

**Тестирование** – ряд мероприятий, связанных с различного рода испытаниями объекта тестирования с целью установления соответствия или несоответствия его характеристик определенным требованиям и выявления дефектов. Дефектами, в свою очередь, могут быть как ошибки в работе, так и неприемлемое качество функционирования в определенных условиях эксплуатации.

Виды тестирования на этапе создания программного продукта классифицируют по преследуемым этим процессом целям. **Функциональное тестирование** - проверка корректного выполнения программой заложенных в нее функций. **Нефункциональное тестирование** - все прочие виды испытаний, такие как тестирование производительности в различных режимах эксплуатации, тестирование эргономики пользовательского интерфейса, тестирование отказоустойчивости и т.д.

**Виды функционального тестирования:**

* **Компонентное (модульное) тестирование.** Тестирование отдельных компонентов программного продукта, сфокусированное на их специфике, назначении и функциональных особенностях.
* **Интеграционное тестирование.** Данный вид тестирования проводится после компонентного тестирования и направлен на выявление дефектов взаимодействия различных подсистем на уровне потоков управления и обмена данными.

**Билет 5, 15, 25.**

**Предназначение инструкции пользователя. Подготовить инструкцию пользователя к модулю, реализованному в ходе курсового проектирования.**

Инструкция (Руководство) пользователя — документ, назначение которого — предоставить людям помощь в использовании некоторой системы. Документ входит в состав технической документации на систему и, как правило, подготавливается техническим писателем.

Разделы руководства пользователя:

1. Введение.

2. Назначение и условия применения.

3. Подготовка к работе.

4. Описание операций.

5. Аварийные ситуации.

6. Рекомендации по освоению.

**В разделе "Введение" указывают:**

1. область применения;

2. краткое описание возможностей;

3. уровень подготовки пользователя;

4. перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю.

**В разделе "Назначение и условия применения" указывают:**

1. виды деятельности, функции, для автоматизации которых предназначено данное средство автоматизации;

2. условия, при соблюдении (выполнении, наступлении) которых обеспечивается применение средства автоматизации в соответствии с назначением (например, вид ЭВМ и конфигурация технических средств, операционная среда и общесистемные программные средства, входная информация, носители данных, база данных, требования к подготовке специалистов и т. п.).

**В разделе "Подготовка к работе" указывают:**

1. состав и содержание дистрибутивного носителя данных;

2. порядок загрузки данных и программ;

3. порядок проверки работоспособности.

**В разделе "Описание операций" указывают:**

1. описание всех выполняемых функций, задач, комплексов задач, процедур;

2. описание операций технологического процесса обработки данных, необходимых для выполнения функций, комплексов задач (задач), процедур.

**В разделе "Аварийные ситуации" указывают:**

1. действия в случае несоблюдения условий выполнения технологического процесса, в том числе при длительных отказах технических средств;

2. действия по восстановлению программ и/или данных при отказе магнитных носителей или обнаружении ошибок в данных;

3. действия в случаях обнаружении несанкционированного вмешательства в данные;

4. действия в других аварийных ситуациях.

6. Рекомендации по освоению

**В разделе "Рекомендации по освоению" указывают** рекомендации по освоению и эксплуатации, включая описание контрольного примера, правила его запуска и выполнения.

**Билет 6, 16.**

**Понятие математической и логической моделей программного продукта. Обосновать выбор математической и логической модели программного кода модуля, реализованного в ходе курсового проектирования.**

**Математическая модель** (модель задачи) — представление задачи в математически универсальном (формализованном) виде; совокупность зависимостей, описывающих входные (исходные) данные, а также их связи в вычислениях промежуточных и конечных данных (результатов).

Математическая модель предназначена для определения значений всех поименованных данных (объектов задачи) напрямую или аналитически.

**Основные требования к математической модели:**

‒ **адекватность** (максимальное соответствие поставленной задаче);

‒ **универсальность** (возможность описания широкого класса однотипных задач).

**Цель моделирования данных на логическом уровне состоит в** обеспечении разработчика ИС концептуальной схемой базы данных в форме одной модели или нескольких локальных моделей, которые относительно легко могут быть отображены в любую систему баз данных. Логический уровень – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире, например, “Отдел”, “Фамилия сотрудника”. Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логическая модель данных может быть построена на основе другой логической модели, например, модели процессов.

**Логическая модель должна удовлетворять следующим правилам:**

1. Каждая функция в модели должна отражать единственную и четко определенную цель. Имена функций должны определять, что должно быть сделано, а не как сделано.

2. Функции должны соответствовать уровню иерархии, на котором они представлены в модели.

3. Связи между функциями (функциональными блоками модели) должны быть минимизированы.

4. Каждая функция должна разделяться не более, чем на 6 подфункций следующего уровня.

5. В модели не должна присутствовать информация, связанная с последующей реализацией изделия, например, такие понятия, как модуль, файл, запись и т. п.

6. Для каждой функции должны быть указаны входные данные.

7. Каждой функции должен соответствовать список выходных данных (выходных отчетов).

**Билет 7, 17.**

**Методы и технологии отладки программ для ЭВМ. Обосновать выбор методов и технологий отладки программного модуля, реализованного в ходе курсового проектирования.**

**Отла́дка** — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.

**Методы отладки:**

1. **Метод ручного тестирования.**Это - самый простой и естественный способ данной группы. При обнаружении ошибки необходимо выполнить тестируемую программу вручную, используя тестовый набор, при работе с которым была обнаружена ошибка.
2. **Метод индукции.**Метод основан на тщательном анализе симптомов ошибки, которые могут проявляться как неверные результаты вычислений или как сообщение об ошибке. Если компьютер просто «зависает», то фрагмент проявления ошибки вычисляют, исходя из последних полученных результатов и действий пользователя. Полученную таким образом информацию организуют и тщательно изучают, просматривая соответствующий фрагмент программы. В результате этих действий выдвигают гипотезы об ошибках, каждую из которых проверяют. Если гипотеза верна, то детализируют информацию об ошибке,   
   иначе - выдвигают другую гипотезу.
3. **Метод дедукции.**По методу дедукции вначале формируют множество причин, которые могли бы вызвать данное проявление ошибки. Затем анализируя причины, исключают те, которые противоречат имеющимся данным. Если все причины исключены, то следует выполнить дополнительное тестирование исследуемого фрагмента. В противном случае наиболее вероятную гипотезу пытаются доказать. Если гипотеза объясняет полученные признаки ошибки, то ошибка найдена, иначе - проверяют следующую причину.
4. **Метод обратного прослеживания.**Для небольших программ эффективно применение метода обратного прослеживания. Начинают с точки вывода неправильного результата. Для этой точки строится гипотеза о значениях основных переменных, которые могли бы привести к получению имеющегося результата. Далее, исходя из этой гипотезы, делают предложения о значениях переменных в предыдущей точке. Процесс продолжают, пока не обнаружат причину ошибки.

**Способы отладки программ**. Для отладки программ обычно применяют три способа:

1) Пошаговая отладка программ с заходом в подпрограммы;

2) Пошаговая отладка программ с выполнением подпрограммы как одного оператора;

3) Выполнение программы до точки останова.

**Билет 8, 18.**

**Понятие алгоритма и формы его представления. Обосновать выбор графического языка спецификаций для представления алгоритма программного модуля, реализованного в ходе курсового проектирования.**

**Понятие алгоритма и формы его представления. Разъяснить порядок действий в алгоритме программного модуля, реализованного в ходе курсового проектирования.**

**Алгоритм** — это описание последовательности действий (план), строгое исполнение которых приводит к решению поставленной задачи за конечное число шагов.

**Общие свойства, которыми обладает любой алгоритм:**

1.**Понятность** — исполнитель алгоритма должен знать, как его выполнять;

2.**Конечность** — выполняемый алгоритм должен приводиться к результату за конечное число шагов;

3.**Дискретность** — любой алгоритм должен состоять из конкретных действий, следующих в определенном порядке;

4.**Массовость** — один и тот же алгоритм можно использовать с различными исходными данными.

5. **Детерминированность** — любое действие алгоритма должно быть строго и недвусмысленно определенно в каждом случае.

**Виды:**

**Линейный алгоритм** — это описание действий, которые выполняются однократно в заданном порядке.

**Разветвляющийся алгоритм** — это алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий.

**Циклический алгоритм** — это описание действий, которые должны повторяться указанное число раз или пока не выполнено заданное условие.

Итак, любой алгоритм можно составить, используя только типовые алгоритмические конструкции. **Формы же представления этих алгоритмов могут быть разными:**

1. **Словесная форма** — это форма описания алгоритма на естественном языке.

2. **Графическая форма** — изображение алгоритма в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

3. **Псевдокоды** — полуформализованные описания алгоритмов на условном алгоритмическом языке, включающем в себя как элементы ЯП, так и фразы естественного языка, общепринятые математические обозначения и др.

4. **Программа** — это запись алгоритма на языке программирования.

**Билет 9, 19.**

**Понятие алгоритма и формы его представления. Разъяснить порядок действий в алгоритме программного модуля, реализованного в ходе курсового проектирования.**

**Алгоритм** — это описание последовательности действий (план), строгое исполнение которых приводит к решению поставленной задачи за конечное число шагов.

**Общие свойства, которыми обладает любой алгоритм:**

1.**Понятность** — исполнитель алгоритма должен знать, как его выполнять;

2.**Конечность** — выполняемый алгоритм должен приводиться к результату за конечное число шагов;

3.**Дискретность** — любой алгоритм должен состоять из конкретных действий, следующих в определенном порядке;

4.**Массовость** — один и тот же алгоритм можно использовать с различными исходными данными.

5. **Детерминированность** — любое действие алгоритма должно быть строго и недвусмысленно определенно в каждом случае.

**Виды:**

**Линейный алгоритм** — это описание действий, которые выполняются однократно в заданном порядке.

**Разветвляющийся алгоритм** — это алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий.

**Циклический алгоритм** — это описание действий, которые должны повторяться указанное число раз или пока не выполнено заданное условие.

Итак, любой алгоритм можно составить, используя только типовые алгоритмические конструкции. **Формы же представления этих алгоритмов могут быть разными:**

1. **Словесная форма** — это форма описания алгоритма на естественном языке.

2. **Графическая форма** — изображение алгоритма в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

3. **Псевдокоды** — полуформализованные описания алгоритмов на условном алгоритмическом языке, включающем в себя как элементы ЯП, так и фразы естественного языка, общепринятые математические обозначения и др.

4. **Программа** — это запись алгоритма на языке программирования.

**Билет 10, 20.**

**Предназначение ТЗ, на написание программного продукта. Подготовить ТЗ, на проектирование модуля (на примере ПМ, реализованного в ходе курсового проектирования).**

**Техническое задание (ТЗ)** — исходный документ, который является основанием для разработки и испытания программы или автоматизированной системы. Техническое задание на программу и программное обеспечение разрабатывается в соответствии с требованиями ГОСТ 19.201-78. Основанием для разработки ТЗ чаще всего является договор.

ТЗ на программу разрабатывается, прежде всего, для тех людей, которые в последствии будут разрабатывать программный продукт. Как и любое другое ТЗ на программу должно быть предельно ясно и не содержать двусмысленные формулировки и должно максимально полно описывать все требования и пожелания Заказчика к создаваемой программе. Состав разделов ТЗ на программу указан всё в том же ГОСТ 19.201–78 (п.1.4).

**Структура технического задания:**

1. введение;
2. основания для разработки;
3. назначение разработки;
4. требования к программе или программному изделию;
5. требования к программной документации;
6. технико-экономические показатели;
7. стадии и этапы разработки;
8. порядок контроля и разработки;
9. в техническое задание допускается включать приложения.