### **Вопрос 1:**

***Опишите и дайте определение "структурного программирования", его суть, достоинства и недостатки.***

***Определение:***

Для любой простой программы можно построить функционально эквивалентную ей структурную программу, т.е. программу, сформированную на основе фиксированного базисного множества, включающего структуру последовательного действия, структуру выбора одного из двух действий и структуру цикла, то есть многократного повторения некоторого действия с проверкой условия остановки повторения.

***Достоинства:***

Базисные алгоритмические конструкции обладают важным свойством – они в точности удовлетворяют определению простой программы, то есть имеют один вход и один выход, что обеспечивает возможность осуществлять их суперпозицию. Любая из трех структур может быть подставлена в остальные или в саму себя. А также позволяет установить простое соответствие между статическим текстом программы и динамическим протоколом ее выполнения

***Недостатки:***

1. Невозможно полностью отказаться от меток и операторов безусловного перехода при создании сложных ПП
2. Сосредоточение на человеческом факторе, а не на логике, что не позволяет
3. Ограниченность в выразительности. Некоторые задачи тяжело выразить в структурах.

### **Вопрос 2:**

***Опишите и дайте определение "Объектно-ориентированного программирования", его суть, достоинства и недостатки***

***Определение***:

Объектно-ориентированное программирование (ООП) — это парадигма программирования, в которой программы организованы вокруг объектов, которые взаимодействуют между собой. Объект представляет собой экземпляр класса, который определяет структуру объекта и его поведение.

***Принципы:***

1. Абстракция
2. Инкапсуляция
3. Иерархия (наследование, агрегация)
4. Полиморфизм
5. Класс

***Достоинства:***

1. Возможность легкой модификации (при грамотном анализе и проектировании)
2. Более легкая расширяемость кода
3. Сокращение количества межмодульных вызовов и уменьшение объемов информации, передаваемой между модулями.
4. Инкапсуляция позволяет скрывать внутренние детали реализации, предоставляя только необходимый интерфейс. Это помогает защищать данные от неправильного использования.

***Недостатки:***

1. Требуется другая квалификация для разработчиков
2. Резко увеличивается время на анализ и проектирование систем
3. Увеличение времени выполнения программы
4. Для некоторых видов задач, таких как некоторые алгоритмы и вычисления, ООП может быть менее эффективным.

### **Вопрос 3:**

***Что такие «нисходящая» и «восходящая» разработка программного обеспечения?***

***Нисходящее программирование*** — это методология разработки программного обеспечения, которая начинается с общих параметров и переходит к детализации и реализации.

***Восходящее программирование***предполагает разработку программы, начиная с низкоуровневых деталей, и постепенно объединяется в более крупные компоненты и системы. Компоненты разрабатываются независимо и объединяются в более крупные

структуры.

### **Вопрос 4:**

***Где и для чего используются Flow-формы и диаграммы Насси-Шнейдермана?***

***Диаграммы Насси-Шнейдермана*** предназначены для визуализации

структуры программы и ее выполнения. Они помогают разработчикам

понять и проектировать алгоритмы и процессы, представляя управляющие

структуры программы, такие как последовательности, ветвления и циклы.

***Flow-формы*** используются для визуализации потоков данных и обмена информацией между различными частями системы и программы.

### **Вопрос 5:**

***Что такое модель «сущность-связь» в нотации Баркера?***

Сущность в нотации Баркера представляется прямоугольником любого размера, содержащим внутри себя:

1. Имя сущности
2. Список имен атрибутов
3. Указатели ключевых атрибутов
4. Указатели обязательных атрибутов

Все связи являются бинарными и представляются линиями с двумя концами (соединяющими сущности), для которых должно быть определено:

1. Имя связи
2. Степень множественности

### **Вопрос 6:**

***Как и для чего используют Диаграммы Джексона и скобочные диаграммы Орра?***

В основе ***диаграмм Джексона*** лежит предположение о том, что структуры данных, так же, как и программ, можно строить из элементов с использованием всего трех основных конструкций: последовательности, выбора и повторения. Каждая конструкция представляется в виде двухуровневой иерархии, на верхнем уровне которой расположен блок конструкции, а на нижнем - блоки элементов.

***Диаграмма Орра*** базируется на том же предположении о сходстве структур программ и данных, что и диаграмма Джексона. Отличие состоит лишь в нотации. Автор предлагает для представления конструкций данных использовать фигурные скобки

### **Вопрос 7:**

***Как называется разбиение предметной области на составляющие? Назовите два вида такого разбиения и их основные элементы.***

Разбиение предметной области на составляющие называется декомпозицией. И для этого существует два типа:

1. Алгоритмическая декомпозиция, где основные элементы – алгоритмы.
2. Объектная декомпозиция, где основными элементами – виды абстракций (классы) и представители этих классов (объекты)

В ***алгоритмической декомпозиции***, при анализе задачи мы пытаемся понять, какие алгоритмы необходимо разработать для ее решения, каковы спецификации этих алгоритмов (вход, выход), и как эти алгоритмы связаны друг с другом.

В рамках ***объектной декомпозиции*** мы пытаемся выделить основные содержательные элементы задачи, разбить их на типы (классы). Далее для каждого класса абстракций мы определяем его свойства (данные) и поведение (операции), а также, как эти классы абстракций взаимодействуют друг с другом.

### **Вопрос 8:**

***Что такое «Интерфейс» и какую роль он играет для спецификации?***

***Интерфейс (interface) –*** это набор операций, используемый для описания сервиса класса или компонента.

***Роль в спецификации:***

1. Определение функций. Интерфейс специфирует, какие функции или

методы представляются компонентом, и как их вызвать.

1. Описание параметров. Он указывает типы и порядок параметров,

которые ожидаются или возвращаются функциями или методами.

### **Вопрос 9:**

***Дайте определение всем четырем видам отношений между элементами***

***модели UML?***

В UML используются четыре основных типа отношений:

* Зависимость. ***Определение:*** Отношение зависимости указывает на то, что изменение независимой сущности каким-то образом влияет на зависимую сущность.
* Ассоциация. ***Определение***: Отношение ассоциации имеет место, если одна сущность непосредственно связана с другой (или с другими — ассоциация может быть не только бинарной).
* Обобщение. ***Определение:*** Отношение между двумя сущностями, одна их которых является частным (специализированным) случаем другой.
* Реализация. ***Определение:*** Отношение реализации указывает, что одна сущность является реализацией другой.

### **Вопрос 10:**

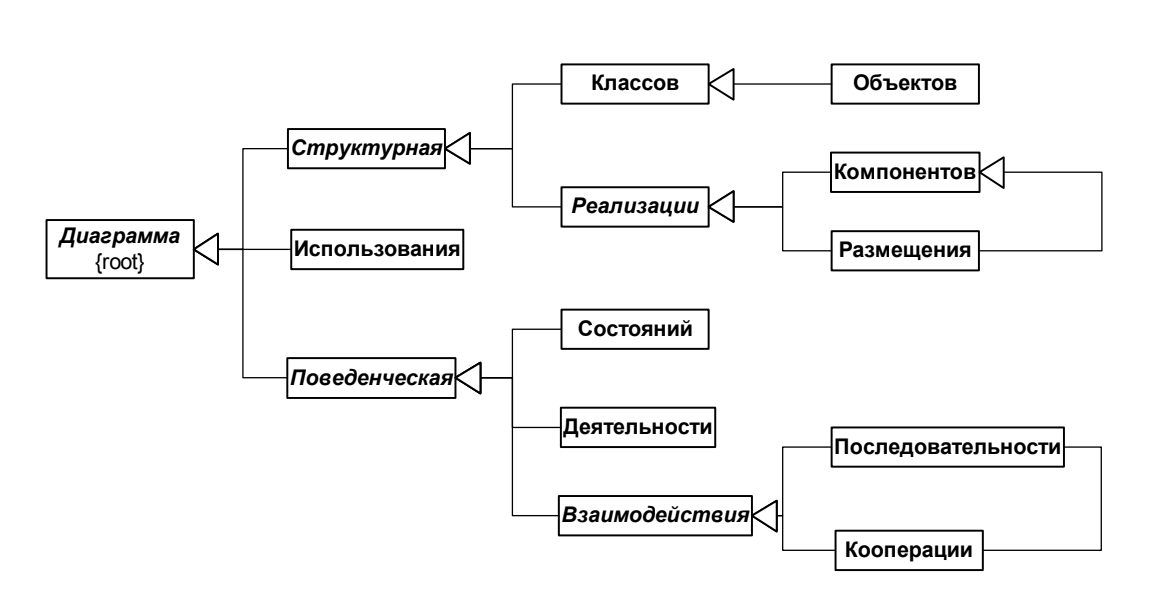
***Опишите иерархию типов диаграмм UML?***

В UML 1.x всего определено 9 канонических типов диаграмм. Ниже перечислены

их названия:

Диаграмма использования

* Диаграмма классов
* Диаграмма объектов
* Диаграмма состояний
* Диаграмма деятельности
* Диаграмма последовательности
* Диаграмма кооперации
* Диаграмма компонентов
* Диаграмма размещения



### **Вопрос 11:**

***Что включает в себя подготовительная работа перед составлением программы?***

1. ***Понимание задачи:***
   1. ***Анализ требований***: внимательно изучите требования к программе.Разберитесь в том, что от вас ожидается, какие функции должны быть реализованы, и какие результаты должны быть достигнуты.
2. ***Создание плана:***
   1. ***Разработка стратегии:*** определите общую стратегию решения задачи. Разбейте ее на подзадачи и определите последовательность выполнения.
   2. ***Оценка ресурсов:*** определите, какие ресурсы потребуются для реализации программы, включая время, трудозатраты и необходимые инструменты.
3. ***Исследование:***
   1. ***Поиск аналогов:*** изучите аналогичные программы или решения, чтобы понять, какие подходы могут быть эффективными. Это может помочь избежать избыточной работы и найти лучшие практики.
   2. ***Изучение технологий***: если задача включает использование новых технологий, изучите их основы и принципы работы.
4. ***Определение архитектуры:***
   1. ***Проектирование структуры:*** определите общую структуру программы, ее компоненты и взаимосвязи между ними.
   2. ***Выбор паттернов проектирования:*** решите, какие паттерны проектирования могут быть полезны для решения задачи.
5. ***Выбор языка программирования и инструментов***:
   1. ***Выбор языка программирования***: Учитывая требования и характер задачи, выберите наилучший язык программирования.
   2. ***Определение инструментов***: Выберите инструменты разработки, среды программирования и другие технические средства.
6. ***Планирование тестирования и отладки:***
   1. ***Определение тестов:*** Создайте план тестирования, который включает в себя тестовые случаи для проверки различных аспектов программы.
   2. ***Стратегия отладки:*** Придумайте стратегию для выявления и исправления ошибок в программе.
7. **Документирование**:
   1. ***Составление документации:*** Подготовьте необходимые документы, такие как техническое задание, комментарии к коду и руководство пользователя.

### **Вопрос 12:**

***Что должно входить в документацию программного средства(Какие постулаты)?***

Документация предусматривает формализованное описание информации, созданной в течение ЖЦ ПО. Данный процесс состоит из набора действий, с помощью которых планируют, проектируют, разрабатывают, выпускают, редактируют, распространяют и сопровождают документы, необходимые для всех заинтересованных лиц, таких, как руководство, технические специалисты и пользователи системы.

### **Вопрос 13:**

***Дайте определение правильности программы. Почему тестирование не показывает правильности программы?***

***Определение:***

Правильность программы обозначает соответствие программы ее спецификации, то есть то, что программа делает, соответствует тому, что от нее ожидается. Программа считается правильной, если она ведет себя в соответствии с требованиями, предъявляемыми к ней, и выполняет свои функции без ошибок.

***Почему тестирование не показывает правильность программы?***

1. ***Неполное покрытие:*** Тестирование, как правило, не может охватить все возможные входные данные и сценарии использования. Даже при

тщательном тестировании остаются некоторые комбинации входных

данных, которые могут быть пропущены.

1. ***Отсутствие спецификации:*** Если у программы нет четкой и полной

спецификации, тестирование может не обнаружить некоторые дефекты, поскольку некорректное поведение может быть непредсказуемым и неизвестным для тестировщиков.

1. ***Логические ошибки:*** Тестирование обычно ориентировано на обнаружение синтаксических ошибок и ошибок в выполнении. Однако логические ошибки, связанные с неправильной реализацией алгоритмов или неверными логическими операциями, могут быть упущены.

### **Вопрос 14:**

***Что такое надежность программного средства?***

***Определение:***

***Надежность ПС*** - способность программного продукта безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени с достаточно большой вероятностью

### **Вопрос 15:**

***В чем проявляется сложность разработки программных средств?***

Большинство современных программных систем объективно очень сложны. Эта сложность обусловливается многими причинами, главной из которых является логическая сложность решаемых ими задач. А также количество ресурсов, связей и компонентов, которые требуются для решения какой-либо задачи, растет нелинейно по мере роста размеров проекта.

- сложность формального определения требований к программным системам;

* Во-первых, при определении требований необходимо учесть большое количество различных факторов. Во-вторых, разработчики программных систем не являются специалистами в автоматизируемых предметных областях, а специалисты в предметной области, как правило, не могут сформулировать проблему в нужном ракурсе

- отсутствие удовлетворительных средств описания поведения дискретных систем с большим числом состояний при недетерминированной последовательности входных воздействий;

* Это приводит к ранней детализации операций в процессе создания программного обеспечения и увеличивает объём описаний разрабатываемых продуктов, который, как правило, превышает сотни тысяч операторов языка программирования. Средств же, позволяющих детально описывать поведение сложных дискретных систем на более высоком уровне, чем универсальный язык программирования, не существует

- коллективная разработка;

* Из-за больших объёмов проектов разработка программного обеспечения ведется коллективом специалистов. Работая в коллективе, отдельные специалисты должны взаимодействовать друг с другом, обеспечивая целостность проекта, что при отсутствии удовлетворительных средств описания поведения сложных систем, упоминавшемся выше, достаточно сложно. Причём чем больше коллектив разработчиков, тем сложнее организовать процесс работы

- необходимость увеличения степени повторяемости кодов.

* На сложность разрабатываемого программного продукта влияет и то, что для увеличения производительности труда компании стремятся к созданию библиотек компонентов, которые можно было бы использовать в дальнейших разработках. Однако в этом случае компоненты приходится делать более универсальными, что в конечном итоге увеличивает сложность разработки

### **Вопрос 16:**

***Раскройте специфические особенности разработки программных средств.***

Выделяют несколько видов особенностей:

1. ***Разработка носит творческий характер.*** (На каждом этапе нужно принимать обдуманные решения, производить какой-либо выбор, а не руководствоваться какой-либо последовательностью механических действий. )
2. ***Особенность разрабатываемого продукта.*** (ПС представляет собой совокупность некоторых текстов (статических объектов). Смысл же (семантика) этих объектов выражается процессами обработки данных и действиями пользователей по запуску этих процессов (т.е. является динамическим))
3. ***Продукт разработки (или ПС) при своём использовании не расходуется и не расходует используемых ресурсов.***

### **Вопрос 17:**

***Опишите процессы абстракции и конкретизации при блочно-иерархическом подходе создания сложных систем.***

Существует несколько типов блочно-иерархическом подхода:

1. ***«целое-часть».*** Такой тип предполагает, что каждую подсистему разделить на подсистемы и т.д. до самого нижнего «элементарного» уровня, причём выбор уровня, компоненты которого следует считать элементарными, остаётся за исследователем
2. ***Простое-сложное*** (иерархия развития (усложнения) систем в процессе эволюции. В этой иерархии любая функционирующая система является результатом развития более простой системы. Именно данный вид иерархии реализуется механизмом наследования объектно-ориентированного программирования)

### **Вопрос 18:**

***Перечислите принципы блочно-иерархического подхода создания сложных систем.***

В основе ***блочно-иерархического подхода*** лежат декомпозиция и иерархическое упорядочение. Важную роль играют также следующие принципы:

1. Формализация – строгость методического подхода;
2. Повторяемость – необходимость выделения одинаковых блоков для удешевления и ускорения разработки;
3. Локальная оптимизация – оптимизация в пределах уровня иерархии;
4. Непротиворечивость – контроль согласованности элементов между собой;
5. Полнота – контроль на присутствие лишних элементов. Совокупность языков моделей, постановок задач, методов описаний некоторого иерархического уровня принято называть уровнем проектирования.

### **Вопрос 19:**

***Какие комплексные показатели качества программных средств вы знаете?***

1. ***Показатели надёжности*** программных средств (ПС) (устойчивость функционирования, работоспособность);
2. ***Показатели сопровождения*** (структурность, простота конструкции, наглядность, повторяемость);
3. ***Показатели удобства применения*** (лёгкость освоения, доступность эксплуатационных программных документов, удобство эксплуатации и обслуживания);
4. ***Показатели эффективности*** (уровень автоматизации, временная эффективность, ресурсоёмкость);
5. ***Показатели универсальности*** (гибкость, мобильность, модифицируемость);
6. ***Показатели корректности*** (полнота реализации, согласованность, логическая корректность, проверенность).

### **Вопрос 20:**

***Опишите качественные характеристики программных средств по ГОСТу Р-ИСО9126.***

1. ***Функциональные возможности*** (Functionality) – набор атрибутов, относящихся к сути набора функций и их конкретным свойствам. Функциями являются те, которые реализуют установленные или предполагаемые потребности;
2. ***Надёжность*** (Reliability) – набор атрибутов, относящихся к способности программного обеспечения сохранять свой уровень качества функционирования при установленных условиях за установленный период времени;
3. ***Практичность*** (Usability) – набор атрибутов, относящихся к объему работ, требуемых для использования и индивидуальной оценки такого использования определенным или предполагаемым кругом пользователей;
4. ***Эффективность*** (Efficiences) – набор атрибутов, относящихся к соотношению между уровнем качества функционирования программного обеспечения и объемом используемых ресурсов при установленных условиях;
5. ***Сопровождаемость*** (Maintainability) – набор атрибутов, относящихся к объему работ, требуемых для проведения конкретных изменений (модификаций);
6. ***Мобильность*** (Portability) – набор атрибутов, относящихся к способности программного обеспечения быть перенесенным из одного окружения в другое.

### **Вопрос 21:**

***Перечислите требования, предъявляемые к модулям при реализации декомпозиции для разрабатываемого программного обеспечения.***

Первоначально к модулям предъявлялись следующие требования:

1. ***Отдельная компиляция***
2. ***Одна точка входа***
3. ***Одна точка выхода***
4. ***Соответствие принципу вертикального управления***
5. ***Возможность вызова других модулей***
6. ***Небольшой размер (до 50 – 60 операторов языка)***
7. ***Независимость от истории вызовов***
8. ***Выполнение одной функции***

### **Вопрос 22:**

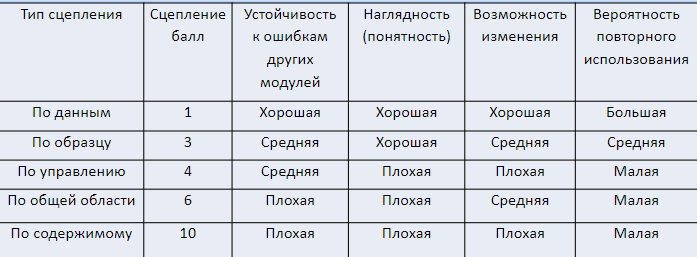
***Перечислите типы сцепления модулей и поясните, почему не все из них используются?***

Сцепление является мерой взаимозависимости модулей, которая определяет, насколько хорошо модули отделены друг от друга. Модули независимы, если каждый из них

не содержит о другом никакой информации. Чем больше информации о других модулях хранит модуль, тем больше он с ними сцеплен.

Различают пять типов сцепления модулей:

* по данным
* по образцу
* по управлению
* по общей области данных
* по содержимому



Допустимыми считают первые три типа сцепления, так как использование остальных приводит к резкому ухудшению технологичности программ. Как правило, модули сцепляются между собой несколькими способами. Учитывая это, качество программного обеспечения принято определять по типу сцепления с худшими характеристиками. Так, если использовано сцепление по данным и сцепление по управлению, то определяющим считают сцепление по управлению.

### **Вопрос 23:**

***В чём суть связности программных модулей и какие бывают виды связностей?***

***Определение:***

Связность – мера прочности соединения функциональных и информационных объектов внутри одного модуля. Если сцепление характеризует качество отделения модулей, то связность характеризует степень взаимосвязи элементов, реализуемых одним модулем. Размещение сильно связанных элементов в одном модуле уменьшает межмодульные связи и, соответственно, взаимовлияние модулей. В то же время помещение сильно связанных элементов в разные модули не только усиливает межмодульные связи, но и усложняет понимание их взаимодействия. Объединение слабо связанных элементов также уменьшает технологичность модулей, так как такими элементами сложнее мысленно манипулировать.

***Виды(Различают следующие виды связности):***

* функциональную
* последовательную
* информационную (коммуникативную)
* процедурную
* временную
* логическую
* случайную

### **Вопрос 24:**

***Приведите определения и различие двух типов библиотек?***

***Различают библиотеки ресурсов двух типов:***

* ***Библиотеки подпрограмм*** реализуют функции, близкие по назначению, например библиотека графического вывода информации. Связность подпрограмм между собой в такой библиотеке – логическая, а связность самих подпрограмм функциональная, так как каждая из них обычно реализует одну функцию.
* ***Библиотеки классов*** реализуют близкие по назначению классы. Связность элементов класса информационная, связность классов между собой может быть функциональной для родственных или ассоциированных классов и логической для остальных.

### **Вопрос 25:**

***Для чего используют «модули-заглушки» при нисходящем подходе разработки ПО?***

При использовании нисходящего подхода разработки компоненты нижних, ещё не реализованных уровней заменяют специально разработанными отладочными модулями – «заглушками», что позволяет тестировать и отлаживать уже реализованную часть.