**Обязательные вопросы:**

1. Что такое инкапсуляция?

**Ответ:** **Инкапсуляция** – это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали реализации от пользователя. Инкапсуляция неразрывно связана с понятием интерфейса класса. По сути, всё то, что не входит в интерфейс, инкапсулируется в классе.

2. Что такое наследование? Приведите пример

**Ответ: Наследование** – механизм языка программирования, позволяющий создавать новые классы с приобретением реализации другого класса. В отличие от агрегирования, где один объект является полем другого объекта, при наследовании один класс (дочерний, производный) является частным, расширенным вариантом другого класса (родительского, базового). При этом все поля и методы базового класса становятся полями и методами дочернего класса.

3. Что такое полиморфизм? Приведите пример

**Ответ: Полиморфизм** — одна из трех основных парадигм ООП. Если говорить кратко, полиморфизм — это способность объекта использовать методы производного класса, который не существует на момент создания базового.

**Вопросы первой группы:**

1. Почему плохо дублирование кода?

**Ответ**: ухудшает восприятие кода, можно легко запутаться. Также усложняет исправление багов или внедрение новых свойств кода.

2. Почему важно единообразное оформление кода?

**Ответ**: для того, чтобы себе и другим разработчикам было проще ориентироваться в коде.

3. Почему нельзя использовать русский язык в названиях переменных, методов и классов?

**Ответ**: русский язык не воспринимается ЯП и IDE.

4. Почему нельзя использовать транслитерацию в названиях переменных, методов и классов?

**Ответ**: это противоречит стандартам (в частности RSDN), затрудняет работу с кодом, а также может быть непонятным для зарубежных разработчиков.

5. Почему важно придерживаться правила "один класс в одном файле"?

**Ответ**: для наглядности. Чтобы легко было найти необходимый класс по названию файла, а не искать его в других файлах.

6. Почему важно сразу же исправлять грамматические ошибки в названиях переменных, методов, классов?

**Ответ**: следуя принципам ООП, разработчики могут интуитивно описывать объект класса, применять его методы и т.д.. Предположительно, они будут использовать грамматически правильные названия и смогут быстро сориентироваться в классе.

7. Что такое указатель?

**Ответ: Указатель** — это переменная, значением которой является адрес (ячейка) памяти

8. Что такое разыменование?

**Ответ:** Операция разыменования указателя представляет выражение в виде \*имя\_указателя. Эта операция позволяет получить объект по адресу, который хранится в указателе.

9. Почему в указателе одного типа нельзя хранить адрес переменной другого типа, хотя формат адресов для переменных разных типов одинаковый?

**Ответ**: потому что в С++ присутствует строгая типизация объектов и в указателе одного типа можно хранить только переменную этого же типа или переменные дочерних классов

10. Что такое исключение?

**Ответ**: Говоря о об исключениях в С++, рассмотрим одну из таких нестандартных ситуаций — деление на 0. Представьте, что мы работаем в программе, где вручную вносим много числовых данных. Далее, в результате каких-то расчетов программы, значение одной из переменных станет равным 0. А в следующей строке кода это значение выступает делителем. Конечно, это действие приведет к неизбежному закрытию программы и выходу из нее. И вы можете себе представить свое состояние, когда полчаса вводите данные, работая в этой программе, и на каком-то этапе она просто перестает работать и закрывается, а все данные пропадают.

11. Что делает оператор throw?

**Ответ**: В C++ оператор throw используется, чтобы сигнализировать о возникновении исключения или ошибки (аналогия: арбитр свистит). Сигнализирование о том, что произошло исключение, называется генерацией исключения (или ещё «выбрасыванием исключения»).

12. Что делают операторы try-catch?

**Ответ**: Фактически, обработка исключений — это работа блока(ов) catch. Ключевое слово catch используется для определения блока кода (так называемого «блока catch»), который обрабатывает исключения определённого типа данных. Блоки try и catch работают вместе. Блок try обнаруживает любые исключения, которые были выброшены в нём, и направляет их в соответствующий блок catch для обработки. Блок try должен иметь, по крайней мере, один блок catch, который находится сразу же за ним, но также может иметь и несколько блоков catch, размещенных последовательно (друг за другом). Как только исключение было поймано блоком try и направлено в блок catch для обработки, оно считается обработанным (после выполнения кода блока catch), и выполнение программы возобновляется.

13. Что такое массив?

**Ответ**: Массив – это переменная, которая является совокупностью компонентов одного типа.

14. В чем отличие передачи указателя в функцию и передачи ссылки в функцию?

**Ответ**: В принципе, технически разница между указателем и ссылкой лишь в том, что саму ссылку нельзя изменить (а указатель можно). Кроме этого, разница ещё синтаксическая: с ссылкой вы обращаетесь как будто это переменная, а с указателем нужно его правильно получать/разыменовывать.

15. Как передать массив в функцию?

**Ответ**: необходимо передать адрес массива. Когда массив используется в качестве аргумента функции, передается только адрес массива, а не копия всего массива. При вызове функции с именем массива в функцию передается указатель на первый элемент массива. (Надо помнить, что в С имена массивов без индекса - это указатели на первый элемент массива.) Параметр должен иметь тип, совместимый с указателем.

16. Что такое перечисление?

**Ответ**: **Перечисление** (или ещё «перечисляемый тип») — это тип данных, где любое значение (или ещё «перечислитель») определяется как символьная константа. Объявить перечисление можно с помощью ключевого слова enum.

17. В чем преимущества перечисления перед использованием строк?

**Ответ**: Перечисляемые типы невероятно полезны для документации кода и улучшения читабельности. Они помогают избежать «магических чисел».

18. Что такое структура?

**Ответ**: Структура – пользовательский тип данных, объединяющий различные типы данных, у которых есть некоторая логическая взаимосвязь

19. Что такое динамическая память?

**Ответ**: Динамическая память выделяется в куче (heap) и фактически ограничивается только размером оперативной памяти.

20. В чем преимущества использования динамической памяти?

**Ответ:** Динамическая память выделяется в куче (heap) и фактически ограничивается только размером оперативной памяти. Можно менять размер данных в процессе исполнения программы.

**Вопросы второй группы:**

1. Что такое класс?

**Ответ: Класс** — в объектно-ориентированном программировании, представляет собой шаблон для создания объектов, обеспечивающий начальные значения состояний: инициализация полей-переменных и реализация поведения функций или методов.

В реальном мире можно найти множество индивидуальных объектов одного и того же типа. Могут существовать сотни и тысячи велосипедов одной и той же модели. Каждый из этих велосипедов создан по одним чертежам (шаблонам), и следовательно имеет одинаковые составляющие и свойства. В терминах ООП, можно сказать, что велосипеды являются экземплярами (instances) класса «велосипед». И класс выступает как шаблон для изготовления экземпляров.

2. Что такое объект класса?

**Ответ**: В C++ переменная класса называется экземпляром или объектом класса. Точно так же, как определение переменной фундаментального типа данных (например, int x) приводит к выделению памяти для этой переменной, так же и создание объекта класса (например, DateClass today) приводит к выделению памяти для этого объекта.

3. Что такое модификатор доступа?   
**Ответ**: Модификаторы доступа позволяют задать допустимую область видимости для членов класса, то есть контекст, в котором можно употреблять данную переменную или метод.

4. Что такое сокрытие реализации?

**Ответ**: ограничение доступа методам, переменным класса.

5. Что такое интерфейс класса? **Ответ**: public методы, через которые можно взаимодействовать с объектом.

6. Что такое реализация класса?

**Ответ**: написание его переменных и методов.

7. Что такое состояние класса?

**Ответ**: значения переменных класса

8. Что такое поведение класса?

**Ответ**: выполнение методов класса

9. Что такое сеттеры? Для чего они нужны?

**Ответ**: Название происходит от английского set — “устанавливать” (т.е. “метод для установки значения поля”). Задают значения переменным объекта.

10. Что такое геттеры? Для чего они нужны?

**Ответ**: Название происходит от английского “get” — “получать” (т.е. “метод для получения значения поля”). Отдают значения переменных объекта.

11. Почему поля класса должны быть закрыты?

**Ответ**: для безопасности значений переменных объекта. В случае неправильного изменения значения могут быть ошибки в результате выполнения программы.

12. Что такое конструктор класса?

Ответ: **конструктор** представляет функцию, которая может принимать параметры и которая должна называться по имени класса.

13. Что такое конструктор по умолчанию?

Ответ: Конструктор, который не имеет параметров (или имеет параметры, все из которых имеют значения по умолчанию), называется конструктором по умолчанию. Он вызывается, если пользователем не указаны значения для инициализации.

14. Что такое конструктор копирования?

Ответ: Конструктор копирования нужен нам для того, чтобы создавать «реальные» копии объектов класса, а не побитовую копию объекта. Иногда это принципиально важно. Такую «реальную» копию объекта надо создавать в нескольких случаях:

* когда мы передаем объект в какую-либо функцию в виде параметра;
* когда какая-либо функция должна вернуть объект класса в результате своей работы;
* когда мы в главной функции один объект класса инициализируем другим объектом класса.

<http://cppstudio.com/post/9903/>

15. Что такое деструктор класса? В каких случаях в классе необходимо определять деструктор?

Деструктор выполняет освобождение использованных объектом ресурсов и удаление нестатических переменных объекта. По сути деструктор - это функция, которая называется по имени класса (как и конструктор) и перед которой стоит тильда (~). Деструктор не имеет возвращаемого значения и не принимает параметров. Каждый класс может иметь только один деструктор.

* Деструктор автоматически вызывается, когда удаляется объект. Удаление объекта происходит в следующих случаях:
* когда завершается выполнение области видимости, внутри которой определены объекты
* когда удаляется контейнер (например, массив), который содержит объекты
* когда удаляется объект, в котором определены переменные, представляющие другие объекты
* динамически созданные объекты удаляются при применении к указателю на объект оператора delete

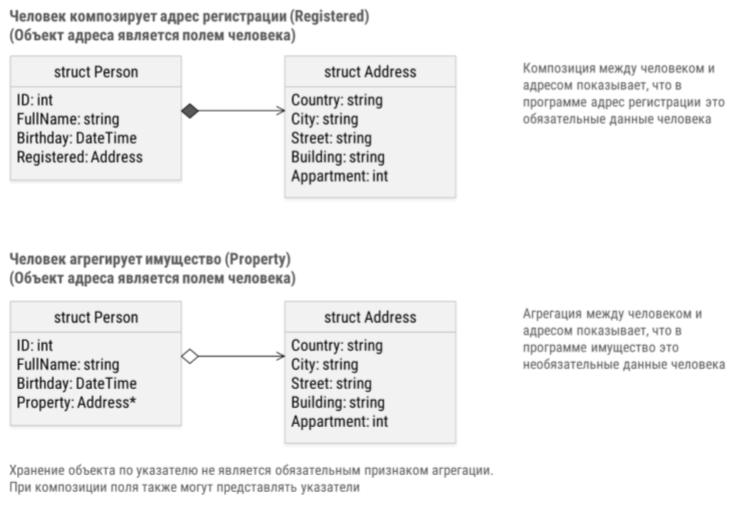
16. Что такое агрегация? Приведите пример

**Ответ**: **Агрегация** – связь между двумя объектами "часть-целое", при котором время жизни объекта-"части" отличается от времени жизни объекта-"целого".

17. Что такое композиция? Приведите пример

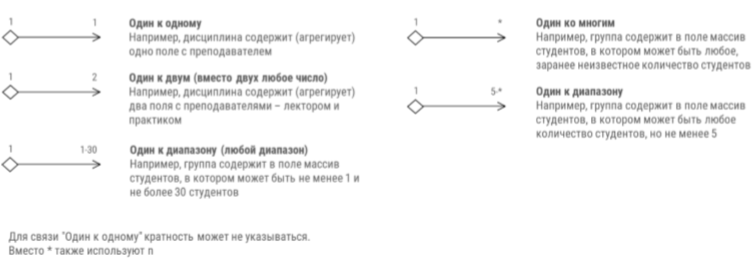
**Ответ: Композиция** – связь между двумя объектами "часть-целое", при котором время жизни объекта-"части" совпадает со временем жизни объекта-"целого"

18. Как обозначаются агрегация и композиция на диаграммах классов?

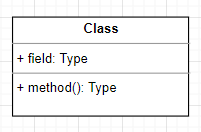


19. Что такое кратность связи? Какие бывают кратности?

**Ответ:** Кратность (в некоторых источниках также называется мощностью или кардинальностью) показывает количественное соотношение объектов между собой, например, "один к одному", "один ко многим", "один к двум" и т.д. На диаграмме кратность обозначается в виде двух чисел, указанных на концах стрелки:



20. Как отображается класс на диаграмме классов?

**Вопросы третьей группы:**

1. Что такое использование? Приведите пример

**Ответ**: В C++ класс, который порождает все остальные классы, называется базовым классом, остальные классы наследуют его свойства, одновременно обладая собственными свойствами. Такие классы называются производными.

2. Что такое указатель на базовый класс? В чем его особенности?

**Ответ**: В С++ указатель на базовый класс может указывать на объект производного класса, полученного из этого базового класса.

3. Почему нельзя хранить в указателе на дочерний класс объект базового класса?

- Дело в том, что наследование не является двусторонней связью, эта связь означает «производный от» (любой дочерний класс – «производный от» своего базового класса), но обратной связи между базовым и производным классами нет. Т.е., по сути, дочерний класс «является» базовым классом с некоторой дополнительной модификацией, но базовый класс «не является» своим дочерним классом и ничего о нём не знает.

4. Что такое виртуальная функция? Для чего они нужны, приведите пример.

**Ответ**: Виртуальные функции – это методы класса, реализация которых может быть переопределена в дочерних классах. В языке Си++ виртуальные функции обозначаются с помощью модификатора virtual. Дочерний класс не обязан переопределять унаследованную виртуальную функцию – он может использовать готовую реализацию из базового класса. Однако, если необходимо переопределить реализацию, то необходимо:

• объявить в дочернем классе функцию с аналогичной сигнатурой;

• обозначить функцию ключевым словом override (перегрузка, переопределение);

• сделать новую реализацию функции в рамках дочернего класса.

5. Что такое чисто виртуальная функция? Для чего они нужны, приведите пример.

**Ответ**: Чисто виртуальные функции – это виртуальные функции, которые не имеют реализации в базовом классе. Синтаксис чисто виртуальных функций выглядит следующим образом:

double virtual GetArea() = 0;

6. Перечислите все существующие связи между классами, начиная от самой сильной к самой слабой.

**Ответ**: Наследование, полиморфизм, статические классы, агрегирование.

7. Почему наследование при возможности стоит заменять на агрегирование?

**Ответ**: для уменьшения зависимости классов друг от друга.

8. В каких случаях наследование не стоит (или нельзя) заменить на агрегирование?

**Ответ:** 1) для сохранения инкапсуляции класса необходимо использование модификатора protected: при агрегировании невозможно предоставить уникальный доступ к отдельным методам агрегируемого класса, в то время как модификатор доступа protected позволяет создать уникальную область видимости членов класса, ограниченную иерархией наследования.

2) использование наследования для реализации полиморфизма: заменить полиморфизм на любую другую связь, при этом не потеряв значительно в лаконичности исходного кода, невозможно.

9. Объясните принцип вызова конструкторов классов при создании объекта дочернего класса

**Ответ**: при вызове конструктора дочернего класса в конструктор необходимо передать аргументы и для дочернего класса, и для родительского. В cpp файле конструктор дочернего класса вызовет конструктор родительского класса и передаст ему необходимые аргументы.

10. Что такое модификатор доступа protected?

**Ответ**: переменные, методы и т.д. под этим модификатором будут доступны дочерним классам, но не будут доступны извне.

11. Что такое абстрактный класс? В чем его отличие от других классов?

**Ответ**: Классы, имеющие хотя бы одну чисто виртуальную функцию без реализации, называются абстрактными классами. Абстрактные классы предназначены для определения общего интерфейса для дочерних классов, но создание объектов базового класса становится недоступным.

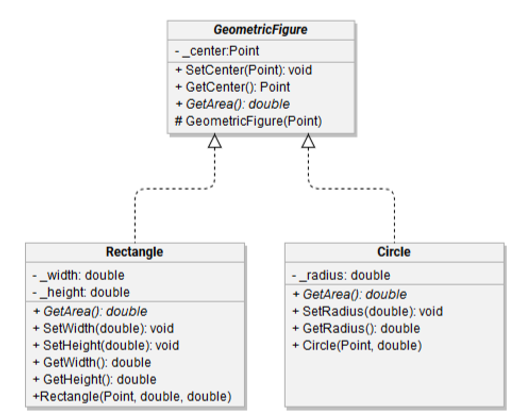
12. Что такое интерфейс? В чем его отличие от других классов?

**Ответ**: Интерфейс – это класс со списком методов у которых нет реализации, т. е. кода заключенного в фигурные скобки.

13. Расскажите алгоритм, по которому из нескольких классов можно выделить базовый класс.

**Ответ**: классы должны иметь одинаковые переменные/методы. Чтобы упростить разработку можно создать базовый класс, вынеся в него одинаковые переменные/методы из других классов.

14. Как обозначается полиморфизм на диаграммах классов?



15. Что такое сервисный класс? Что такое статический класс?

Статический класс – это класс, у которого все переменные, методы и т.д. статические. Они могут использоваться без создания объекта класса в любом месте программы.

Сервисный класс – вид статического класса, который выполняет какие-либо сервисные функции (к примеру, правильность ввода, сортировка массива и т.д.)

16. Почему стоит как можно реже использовать статические методы и классы?

**Ответ**: их состояние можно изменить + это частично нарушает концепцию ООП и больше относится к императивному и процедурному программированию.

17. Как реализовать поле, доступное только на чтение?

**Ответ**: поле должно быть приватным, а отдавать значение должно через геттер.

18. Что такое неизменяемый объект (immutable object)?

**Ответ:** Неизменяемый объект (англ. Immutable object) — в объектно-ориентированном программировании объект, который не можетбыть изменён после своего создания.

Объект может быть неизменяемым как полностью, так и частично. Например, применение директивы const к какому-либо члену класса в C++ делает объект частично неизменяемым. В некоторых случаях объект считается неизменяемым с точки зрения пользователя класса, даже если изменяются его внутренние поля. Как правило, неизменяемый объект получает все внутренние значения во время инициализации, либо значения устанавливаются в несколько этапов, но до того, как объект будет использован.

19. Что такое рефакторинг?

**Ответ**: процесс приведения кода в более читаемый вид, а также в соответствие какому-либо стилю кода

20. В чем цель разделения архитектуры по паттерну "Модель-Вид"?

**Ответ**:

Модель

Под Моделью, обычно понимается часть содержащая в себе функциональную бизнес-логику приложения. Модель должна быть полностью независима от остальных частей продукта. Модельный слой ничего не должен знать об элементах дизайна, и каким образом он будет отображаться. Достигается результат, позволяющий менять представление данных, то как они отображаются, не трогая саму Модель.

Модель обладает следующими признаками:

* Модель — это бизнес-логика приложения;
* Модель обладает знаниями о себе самой и не знает о контроллерах и представлениях;
* Для некоторых проектов модель — это просто слой данных (DAO, база данных, XML-файл);
* Для других проектов модель — это менеджер базы данных, набор объектов или просто логика приложения;

**Представление (View)**

В обязанности Представления входит отображение данных полученных от Модели. Однако, представление не может напрямую влиять на модель. Можно говорить, что представление обладает доступом «только на чтение» к данным.

Представление обладает следующими признаками:

* В представлении реализуется отображение данных, которые получаются от модели любым способом;
* В некоторых случаях, представление может иметь код, который реализует некоторую бизнес-логику.

Примеры представления: HTML-страница, WPF форма, Windows Form.