**Вопрос №1   
Читали ли вы чужой код? Если да, то что это был за код? Опишите свои впечатления от него.**

Да, мне довольно часто приходиться читать чужой код. От рабочих проектов, с которыми мне приходилось работать, до чтения статей. Как код на разных ЯП, так и код вёрстки. Поэтому я умею читать код. Впечатления бывали разные.

**Вопрос №2   
Опишите известные вам языки программирования и их предназначение.**

Языков программирования много. Их разделяют на группы в зависимости от внутреннего устройства, концепций, возможностей платформы и т.д.:

1. Языки высокого уровня: Java, C#, Python, JavaScript и т.д.
2. Языки среднего уровня: C++, Pascal и т.д.
3. Языки низкого уровня: Assembler и т.д.

При том, языки развивались со временем от низкоуровневых к высокоуровневым.

Языки программирования высокого уровня обладают высоким уровнем абстракции, благодаря этому упрощается разработка ПО. Такие языки могут использоваться для создания различных типов приложений: консольные приложения, службы, Desktop, WEB, анализ данных и машинное обучение.

Языки программирования среднего уровня, например, С++ используют для создания высокопроизводительных приложений и ПО взаимодействующее с аппаратными устройствами.

Низкоуровневые языки программирования такие как Assembler работают напрямую с аппаратной частью и памятью компьютера.

Об языках программирования можно говорить довольно много. К тому же, помимо разделения ЯП на уровни, существуют и другие критерии разделения ЯП, например: ООП-язык или процедурный? Со статической типизацией или динамической? и т.д.

**Вопрос №3   
Что такое "компилятор", зачем он нужен и почему некоторые языки обходятся без него?**

Если кратко, то компилятор переводит программу, написанную на языке программирования воспринимаемый человеком, в машинный код, который представляет собой инструкции в виде последовательности нулей и единиц. В результате компиляции создаётся бинарный файл, который исполняется процессором. Пример ЯП, использующий компилятор – С++.

Некоторым ЯП не нужен компилятор, поскольку они используют интерпретатор, который выполняет исходный код программы построчно. В этом случае бинарный файл не создаётся, так как перевод в машинный код и его исполнение происходит построчно. Пример ЯП, использующий интерпретатор – Python.

**Вопрос №4   
Что такое "фреймворк" и для чего он нужен? Приведите примеры известных вам фреймворков.**

Фреймворк – это ПО, который содержит шаблоны, облегчающие разработку и объединение разных компонентов приложения. Фреймворк предоставляет некую готовую функциональность «из коробки», тем самым освобождая разработчика от реализации тривиальных вещей и позволяя сосредоточиться на реализации бизнес-логики приложения.

Примеры: .NET Framework, .NET Core, React/Angular/Vue.js, Django.

**Вопрос №5   
Что за приставка "http://" перед адресами сайтов и почему она всё чаще теперь становится "https://"?**

http - это протокол передачи данных, использующийся в интернете. Раньше использовался для передачи html-страниц, на данный момент умеет передавать данные произвольного типа.

https – расширение протокола http, поддерживающее шифрования в целях повышения безопасности.

Засчёт повышения безопасности https является более предпочтительным протоколом по сравнению с http.

**Вопрос №6   
Самая популярная библиотека для разработки фронтенд-приложений, ReactJS, моделирует логику в виде компонентов.  
  
Если бы вам нужно было на ReactJS разработать страницу профиля ВКонтакте (например,** [**https://vk.com/id1**](https://vk.com/id1)**), на какие компоненты вы бы её разбили? Почему именно так?**

Сайт имеет определённую структуру, которую можно условно разделить на следующие компоненты:

* Шапка сайта
* Боковое меню
* Блок аватарки
* Блок с личными медиа файлами
* Блок с информацией «о себе»
* Отдельный блок с фотографиями
* Лента новостей

Это явно проглядываемые блоки сайта, который служат компонентами для наполнения информацией.

**Вопрос №7   
SqlServer, PostgreSQL, SQLLite, MySQL, Oracle, Microsoft Access - разные базы данных с разным функционалом, которые разрабатываются, в основном, разными компаниями с разным видением своего продукта.  
  
Однако все эти базы используют один и тот же язык запросов - SQL, и не планируют от него отказываться. Как так получается? Что такого в SQL, что он подходит всем этим базам?  
  
А если он такой чудесный, то почему многие другие базы данных, вроде MongoDB или Cassandra, его не используют?**

Дело в том, что SQL является не только языком запросов, но и общепринятым стандартом, укоренившимся крепко и надолго. SQL – является декларативным языком. Это означает, что с помощью него мы описываем, что нам необходимо получить, без уточнения как это получить, то есть отвечает на вопрос «что», а не «как». SQL реализован во многих популярных СУБД, в первую очередь в СУБД, поддерживающих реляционные таблицы, с которыми легче и удобней работать по сравнению с другими структурами данных.

Тем не менее, у реляционных СУБД есть свои ограничения. Так с возрастанием количество записей запросы SQL начинают работать медленней, да и хранения реляционных таблиц занимают много памяти. Поэтому была разработана концепция NoSQL.

В NoSQL отказались от представления данных в виде реляционных таблиц, заменив их другими структурами данных, например, key-value. То есть запись теперь представлена не в виде кортежа, а в виде набора key-value значений. Представителями NoSQL СУБД являются MongoDB, Cassandra и другие.

**Вопрос №8   
Пришлите ссылку на пример вашего кода на C#, за который вам не стыдно. Если кода нет, выполните задание ниже. Оно также поможет, если код есть.**

Ссылка на код, который не стыдно(на самом деле, это не так):

<https://github.com/KhusDM/PersonalAccountOfStudent>

Ссылка на код с задачей:

<https://github.com/KhusDM/Geometry>

**Вопрос №9   
В базе данных MS SQL Server есть продукты и категории. Одному продукту может соответствовать много категорий, в одной категории может быть много продуктов. Напишите SQL запрос для выбора всех пар «Имя продукта – Имя категории». Если у продукта нет категорий, то его имя все равно должно выводиться.**

Там же запрос:

<https://github.com/KhusDM/Geometry>