**Архитектура фон Неймана** - это принцип хранения одновременно в памяти компьютера и команд, и данных. Принципы фон Неймана:

1. Использование двоичной системы счисления. Это нужно для простоты реализации с технической стороны.
2. Программное управление ЭВМ. У нас есть набор команд, которые выполняются друг за другом и с помощью которых мы можем контролировать компьютер.
3. Память компьютера используется не только для хранения данных, но и программ.
4. Ячейки памяти ЭВМ имеют адреса, которые последовательно пронумерованы. Представим память компьютера как одно большое пространство. Оно делится на ячейки (биты), куда записываются данные. Нумерация битов позволяет нам обратиться к любому из них в любой момент времени.
5. Есть возможность условного перехода в процессе выполнения программы. Мы можем использовать функции, процедуры, метки и т.д. для перехода из одного участка кода в другой.

**Устройство архитектуры фон Неймана**  
Компьютер, построенный по принципам архитектуры фон Неймана должен включать в себя: процессор, запоминающее устройство, устройство ввода, устройство вывода. Запоминающее устройство состоит из внешнего запоминающего устройства и оперативного запоминающего устройства. Внешнее запоминающее устройство предназначено для долговременного хранения информации, например, констант, программ и т.д, и её обработки, то есть удаления, редактирования и т.д. Оперативное запоминающее устройство (ОЗУ) используется для хранения и обработки данных, которые задействованы в исполнении какой-либо запущенной программы, после чего память очищается.

**Почему в компьютерах и другой электронике используется двоичная система?**  
Причина заключается в простоте и соответствии двоичной системы одному принципу: чем меньше значений существует в системе, тем проще изготовить отдельные элементы, оперирующие этими значениями. Это значит, что для того, чтобы использовать двоичную систему счисления, достаточно использовать устройства с двумя устойчивыми состояниями. Например, лампочки (горит – не горит) и транзисторы (есть ток – нет тока), и т.д. То есть, у нас есть сигналы, с помощью которых мы можем закодировать что угодно на компьютере. Различая сигналы, компьютер различает символы.

**Структура оперативной памяти**Память состоит из ячеек, каждая из которых имеет свой адрес. Адреса расположены по порядку и записаны в виде двоичных чисел. Как правило, одна ячейка содержит один байт информации. Один байт состоит из 8 бит. То есть каждая ячейка состоит из 8 разрядов. Байт является минимальной единицей, к которой мы можем обратиться. Комбинации из нескольких (как правило, четырёх или восьми) байтов называются словами. Слова иногда используются командами в качестве единицы для обращения.

**В чем заключается задача программиста?**Задача программиста заключается в преобразовании задания, поставленном на естественном языке, к тому виду, в котором оно будет понятно компьютеру.

**Виды языков программирования**  
Программисты пишут код на языке программирования. Их существует большое множество. Но их можно распределить по шкале «низкоуровневые-высокоуровневые».

Низкоуровневый язык программирования — язык программирования, близкий к языку единиц и нулей. Низкоуровневые языки больше понятны компьютеру, чем человеку.  
Высокоуровневый язык программирования — язык программирования, который больше понятен человеку, чем компьютеру.

**IDE**  
Давайте представим, что у нас есть задание написать доклад. Скорее всего, мы будем его печатать в программе «MS Office Word». Программисты тоже пишут код в специальных программах - в IDE. Это интегрированная среда разработки, т.е. программа-редактор, которая помогает разработчику создавать программное обеспечение. IDE представляет собой комплекс инструментов — она включает в себя текстовый редактор, транслятор, средства автоматизации сборки, отладчик, и др. Большинство современных IDE позволяет работать сразу с несколькими языками программирования.

**Транслятор.** Транслятор - это программа, которая преобразует код на языке программирования в код машинный (двоичный). Существует два вида трансляторов: компиляторы и интерпретаторы.

**Компилятор.** Компилятор - это транслятор, который делает весь перевод задачи с естественного языка в машинный код за один подход. Первый компилятор был придуман в 1952 году программисткой по имени Грейс Хоппер. Он был создан таким образом, что пользователь мог пользоваться программой, но не мог прочитать её исходный код.

**Интерпретатор.** Интерпретатор - это транслятор, который делает построчный перевод кода с языка программирования в машинный код. Первый интерпретатор был создан в 1958 году программистом по имени Стив Рассел. Интерпретатор позволяет создавать различные многоступенчатые команды.

**Структура программы**  
Каждая программа на С++ включает в себя функцию main. Основное тело программы содержится именно в ней, сама программа выполняется начиная с функции main. Такая функция должна быть ровно одна.

Функция - это набор операторов, объединённый одним названием. Функция должна возвращать значение.

Пример программы на С++:

#include <iostream>

using namespace std;

int main()

{

cout<<"Hello World";

return 0;

}

**Что такое cout?**   
cout - это выходной поток, т.е. проще говоря с помощью cout мы можем вывести нужную информацию на экран.  
cin - это входной поток, т.е. с помощью cin мы можем ввести с клавиатуры нужные данные.

**Зачем нужно писать std:: ?**std - это пространство имен, где содержатся различные функции и операторы. Существуют и другие пространства имён. Есть два варианта, как их использовать: написать один раз **using namespace <название>;** или перед каждым оператором писать **<название>::**.

**Как правильно?** Предположим, что у нас есть два пространства имён a и b, которые содержат функции, которые названы одинаково, например, ab. Если мы будем писать код, и в самом начале укажем:  
using namespace a;  
using namespace b;  
а потом попытаемся вызвать функцию ab, то компилятор не поймёт, откуда мы хотим взять функцию ab. Поэтому написание a::ab и b::ab считается более правильным.

**Обратите внимание**. После каждой строки (команды/инструкции) в теле функции мы ставим точку с запятой. Она означает завершение команды.