**Информация об уроке**

**Модуль: 1**

**Номер занятия: 3**

**Тип занятия: Лекция**

**Название занятия/ Тема занятия:** Различия и сходства в языках программирования разных уровней; Архитектура сетей

**Цель занятия:** изучить устройство и работу компьютера и его компонентов

**Образовательные результаты:** полное понимание того, как происходят процессы внутри компьютера

**Глоссарий:**

**Конспект занятия**

1. **Приветствие**

Приветствуем вас на новом курсе по Python!

1. **Тема урока и целеполагание**

Сегодня мы пройдем ряд базовых тем, которые зададут весь дальнейший вектор обучения:

1. Различия и сходства в языках программирования разных уровней;
2. Архитектура сетей
3. **Актуализация**

На прошлом занятии мы узнали:

* Архитектура современных компьютеров базируется на принципах фон Неймана
* компьютеры состоят из: процессора, памяти, устройств ввода и вывода, операционной системы
* Процессоры выполняют лишь 4 базовые команды
* Операционная система администрирует работу процессора так, чтобы использовать его максимально эффективно

Все устройства связаны между собой сетью. Этой сетью может быть не только интернет, но и локальная сеть или территориальная.

Все программы создаются программистами с помощью языков программирования. Под каждую задачу найдется несколько подходящих языков, которые в полной мере позволят своим функционалом решить поставленную задачу.

Между собой языки программирования имеют различия, которые как раз определяют тот спектр задач, который можно решить с их помощью.

1. **Основное содержание**

**1. Различия и сходства в языках программирования разных уровней.**

Сначала определимся с тем, что такое программирование. Программа состоит из кода, который управляет отдельными элементами компьютера, позволяя программистам выполнять задачи.

**Язык программирования** - формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (ЭВМ) под ее управлением.

Языки программирования (ЯП) имеют **уровни** взаимодействия с компьютером:

1. **ЯП низкого уровня** (непосредственно прямое управление памятью, потоком электричества, регистрами и процессором) - Машинные коды (0/1), Assembler
2. **ЯП высокого уровня** (Написание программы словами и конструкциями, после чего текст переводится в машинный код и выполняется компьютером)

Особенности ЯП низкого уровня

**плюсы:**

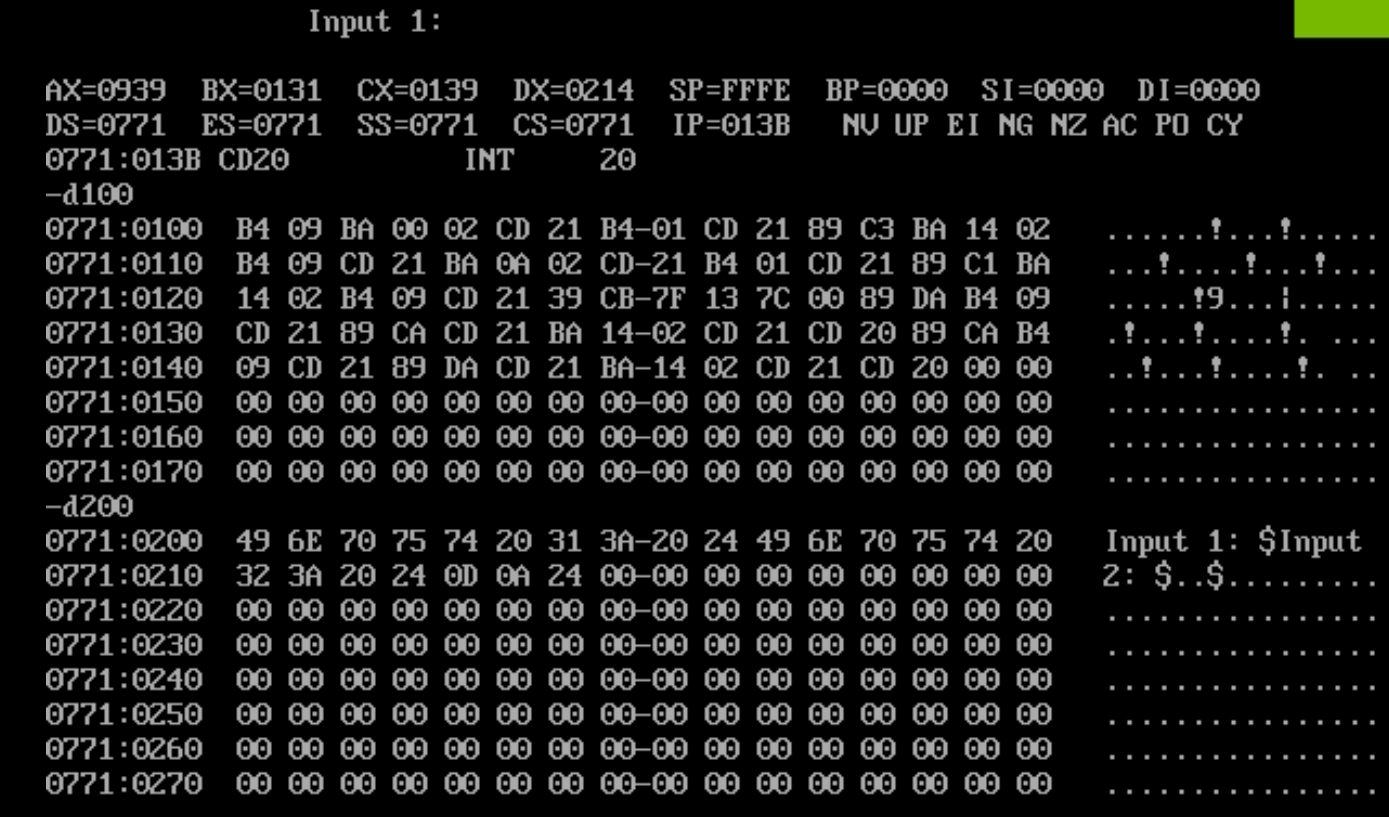
* очень эффективные программы
* компактные программы (занимают мало памяти)
* Полный контроль ресурсов системы

**минусы:**

* громоздкий код
* требуется совершенное понимание работы системы (порог вхождения в отрасль сильно завышен)
* отсутствие библиотек
* несовместимость с другими типами процессоров (программа, написанная для одного процессора, не подходит для другого)

Написание программ в машинных кодах или на Assembler требуют обширных знаний в области физики, электротехники, математического анализа, логики, информатики и алгоритмах. От этого количество специалистов в данной области можно было пересчитать по пальцам. Данные языки до сих пор используются при проектировании ОС и кодировании микросхем.

Написание программ на языке Assembler изучается в ВУЗ-ах. Сам процесс сводится в перебрасывании значений между регистрами микропроцессора с последующей записью результата вручную в ячейки оперативной памяти. Список операций в данном языке довольно скуден: **сложить, умножить** и **разделить** (**вычитание** заменяется суммой с отрицательным вторым операндом)



Языки программирования высокого уровня не требуют настолько обширных знаний поскольку взаимодействие программиста и компонентов компьютера минимизировано. Написание кода в ЯП высокого уровня сводится к написанию текста, который компилятор переводит в машинный (байт) код, игнорируя лишние инструкции в коде и выдавая ошибки компиляции, при попытке запустить некорректную программу. В области высокоуровневых языков программирования специалистов гораздо больше.

Основные виды ЯП:

1. **машинные**
2. **машинно-ориентированные** (Assembler)
3. **алгоритмические** (школьные программы по типу кузнечика)
4. **процедурные** (Fortran, Basic, Pascal, C, …)
5. **объектно-ориентированные** (C++, C#, Java, Python, …)
6. **проблемно-ориентированные** (SQL, Prolog, Perl, HTML, …)
7. **функциональные** (LISP, Scala, Haskell, …)

Этапы появления языков программирования:

1. Fortran (1956) - Formula Translator.
2. Algol (1958) - Algorithmic Language
3. COBOL (1959) - Common Business-Oriented Language
4. Basic (1963)
5. Pascal (1970)
6. C (1970)
7. C++ (1980)
8. Python (1991)
9. Java (1995)
10. C# (2000) - Новый язык от компании Microsoft
11. Golang (2009) - Новый язык от компании Google

Мы с вами заострим внимание на языках высокого уровня. Про сходства мы поговорили, теперь коснемся и различий:

**Различие в типизации**

Языки программирования делятся на 2 группы - **строго типизированные** и с **динамической типизацией**.

**Строгая типизация** означает то, что все переменные должны быть заранее объявлены перед использованием в программе. Компьютер в таком случае выделяет память под данные переменные и освобождает при завершении программы. Такие языки занимают гораздо меньше оперативной и кеш памяти, поскольку размер (в байтах) у переменных не может увеличиться в процессе работы программы. Однако такой тип доставляет неудобства, поскольку к каждой функции, переменной или полям класса надо заранее строго указать тип данных, который будет использовать тот или иной объект. Зачастую бывает такое, что в функцию принимающую числа попадает строка, от чего на этапе сборки кода компилятор будет выдавать ошибку до тех пор, пока она не будет устранена.

**Динамическая типизация** напротив означает то, что тип данных и размер переменных в процессе работы программы может измениться. В таком подходе есть минус - размер программы в оперативной памяти может увеличиться в случае безграмотного написания кода.

**Различие в сборке программы**

Все программы перед тем, как выполниться, проходят этап сборки. За этой отвечают: **компилятор** и **интерпретатор**.

**Компилятор** применяется в языках со статической типизацией. Компилятор переводит программу в машинный код перед ее выполнением. В случае обнаружения ошибки, процесс сборки будет прерван. Нередко данные ошибки приходится искать часами. Такой тип сборки и выполнения программ максимально быстрый, так как мы уже выяснили, что самыми быстрыми программами являются машинные коды. Процесс сборки может быть довольно долгим, однако сама программа будет работать гораздо быстрее.

**Интерпретатор** запускает программу сразу без перевода в машинный код, выполняя его в процессе работы программы, на ходу. У такого подхода есть минус - время выполнения программы увеличивается так как процессор выполняет сразу 2 задачи: перевод из кода в машинный код и выполнение машинного кода. Поскольку заранее интерпретатор не смотрит где есть ошибки в коде, программа будет работать до тех пор, пока не встретит ее, после чего аварийно завершится.

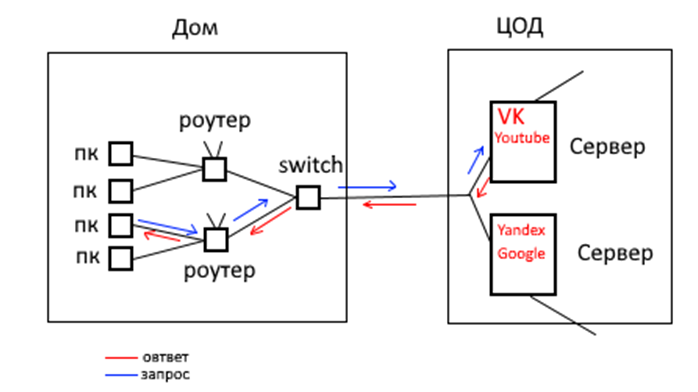
**2. Архитектура сетей.**

**Компьютерная сеть (NET)** - совокупность ЭВМ и других устройств, соединенных линиями связи и обменивающихся между собой данными в соответствии с определенными правилами (протоколами).

Классификация сетей по протяженности линий связи:

* локальные (LAN - Local Area Network)
* региональные (MAN - Metropolitan Area Network)
* Глобальные (WAN - Wide Area Network) - Internet (International NET)

**Сеть Интернет**



**1.** **Программно аппаратное оборудование**

* **Клиенты** (ноутбук, телефон, ПК)
* **Сервер** (устройства, на которых хранится сетевая информация в виде баз данных, которые предоставляют ответы на запросы)
* **Сетевые устройства** (Коммутаторы, модемы, маршрутизаторы)

**2.** **Соединяющие каналы связи**

Сервер = веб. Сервер

**Узел сети** – соединенные между собой устройства, являющиеся частью сети

**Веб Сервер** – это п.о. (программное обеспечение) и компьютер, на котором оно действует, обрабатывающий HTTP запросы / ответы клиента

**HTTP** запросы / ответы могут содержать в себе HTML страницы, файлы, изображения и т.д.

**Клиент** – веб браузер (или другое п.о.), передающий веб серверу запросы на получение ресурсов, обозначенных URL адресами

**Интернет ресурсы** – HTML страницы, фото, видео, файлы …, которые запрашивает клиент

Обмен запросами и ответами между **клиентами** и **веб серверами** происходит посредством протокола **HTTP**

**DNS**

**DNS** (Domain name system) – технология, которая предоставляет браузеру возможность находить конкретный сайт по его имени с помощью DNS-сервера

DNS – фундаментальная технология, которая отвечает за хранение и обработку информации о доменных именах \ адресах. Инструмент используется для преобразования доменных имен в **IP-адреса** в момент отправки пользователем запрос на сервер

**IP-адрес** (Internet Protocol адрес) – уникальный числовой идентификатор устройства, он позволяет узнать, откуда загружается страница нужного сайта

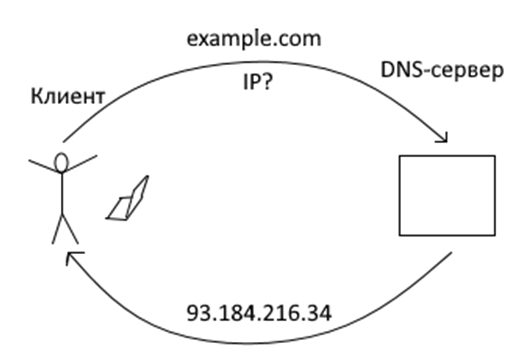
Вывод: **DNS** – телефонная книга, в виде базы данных доменных имён и адресов

Работу **DNS** в качестве базы данных доменных имён и адресов обеспечивает **DNS-сервер**

**DNS-сервер** – оборудование или п.о., с помощью которого предоставляется доступ к системе доменных имен, хранятся данные о соответствии конкретного **IP** соответствующему домену, а также осуществляется **кэширование** информации в виде **IP** и соответствующего ему домена других **DNS-серверов**

**Кэширование** – процесс, при котором элементы сайта или страница целиком сохраняются в КЭШе.

Схема работы DNS-технологии



Пользователь вводит в адресную строку браузера доменное имя, а преобразователь доменных имён обращается к **DNS-серверу**. После получения **IP-адреса**, **DNS-сервер** передает его браузеру пользователя. Затем браузер делает запрос на **сервер** по этому **IP-адресу**, и после получения ответа отображает страницу ресурса.

**HTTP**

**HTTP** (Hyper Text Transfer Protocol) – широко распространённый протокол передачи данных, изначально предназначенный для передачи гипертекстовых документов (то есть документов, которые могут содержать ссылки, позволяющие организовать переход к другим документам)

Протокол **HTTP** предполагает клиент-серверную структуру передачи данных. Также часто используется, как протокол передачи информации для других протоколов прикладного уровня: **SOAP**, **XML-RPC**, **WebDAV**

Задача **HTTP**: обмен данными между **пользовательским** **приложением** и **сервером**

**API** многих программных продуктов подразумевает использование **HTTP** для передачи данных – сами данные при этом могут иметь любой формат (**JSON, XML**)

Как правило, передача данных осуществляется через **TCP\IP-соединения**. Серверное программное обеспечение при этом обычно использует **TCP-порт 80** (Дефолтное значение, если он не указан явно), хотя может использовать любой

**Безопасность**

Сам по себе **HTTP** не предполагает шифрования для передачи информации. Однако существует расширение (**HTTPS**), которое реализует упаковку данных в криптографический протокол **SSL** или **TLS**.

Расширение **HTTPS** (Secure) – использует обычно порт **443**.

Существует также протокол **SPDY** (Speedy – не аббревиатура) – модификация **HTTP,** у которой меньше скорость загрузки и повышена безопасность

**Доменные имена**

**Доменное имя** – адрес сайта, состоящий из символов или цифр, но напрямую не связанный с сервером. **DNS-сервер** напрямую ассоциирует **доменное имя** и **IP-адрес** или другими **доменными именами**.

Пример: Mail.ru – 94.100.180.200

**Доменные имена** в интернете используют несколько уровней (обычно 4) для того, чтобы сделать название краткими

Уровни доменных имен:

1. **Домен 0 уровня** (корневой домен) – обозначается пустым именем, не содержащим никаких символов. Нулевые домены администрирует **ICANN**.

2. **Домен 1 уровня** (верхний) – разбивается на 2 подтипа:

a. Национальный – сообщает принадлежность стране (Россия - **.ru**, **.рф**, **.su**; Беларусь - **.by**…)

b. Родовые – сообщает сферу деятельности (**.com** – коммерческие организации, **.org** – некоммерческие организации, **.law** – лицензированные юристы …)

3. **Домен 3 уровня** (поддомен / субдомен) – например создатель сайта **eda.ru** решил открыть форум **forum.eda.ru**

4. **Домен 4 уровня** – используется очень редко. Пример: **vahitovsky.tat.sudrf.ru** – зарегистрирован в РФ, является сайтом суда, находится в Татарстане, а именно Вахитовский суд

1. **Подведение итогов/Рефлексия**

На этом занятии мы узнали какие бывают языки программирования, в чем их сходства и различия. Что такое сеть и из чего она состоит. Как происходят запросы в сети интернет.