1. Что с собой представляет программа на лиспе?

2. Основные приципы функционального программирования, методологии и языки

## Методологии программирования:

* Императивное – пошаговое изменение сост-я вычислителя(памяти - регистров *вроде*)
* Функциональное – чуть ниже
* Логическое
* Объектно-ориентированное

## Основные идеи функционального программирования:

* Отсутствие понятие «оператор»
* Базовое понятие – «функция»
* Базовая структура – «список»
* Нет четкого разделения на программу и данные

3. Что такое список?

<список> ::= <NIL> | (<s-выражения> . <список>)

хвост

голова

4. Базис языка, определения

## Базовые функции Lisp:

1. Селекторы
   1. car – предназначена для получения первого элемента точечной пары (или же головы списка)
   2. cdr – предназначена для получения второго элемента точечной пары (или же хвоста списка).
2. Конструкторы
   1. cons – создает точечную пару
   2. list – создает список
3. Предикаты
   1. NULL
   2. consp – проверка, является ли списковой ячейкой
   3. listp – проверка, является ли списком
   4. Atom – проверка, является атомом или нет
   5. eql – сравнение по указателям
   6. equal - eql + сравнение списков
   7. Equalp – eql + сравнение чисел

5. Что такое функция

1.Встроенные в ядро Лиспа функции, реализованные в машинных кодах (или на языке реализации ядра Лиспа). Тип SUBR  
2.Реализованные на языке Лисп. тип EXPR   
  
Вызовы функций типа SUBR и типа EXPR не отличаются по форме и выглядят следующим образом:

(Функция Аргумент1 Аргумент2 ... Аргументn)

Функция "знает" сколько аргументов ей требуется. Если количество аргументов оказывается больше или меньше необходимого, возникает ошибка вычисления функции и выдается соответствующее сообщение. Бывают функции с переменным числом аргументов, а бывают и функции без аргументов. Обращение к такой функции выглядит так: (Функция)

абсолютно неверна. Эта запись означает вычисление функции F с двумя аргументами: первый - G второй X! Более логично было бы записать конструкцию F(G(x)) в виде:

Если Лисп-машина, прежде чем вычислять значение функции F, сначала вычислит значение функции G (т.е. значение выражения (G X)) и заменит в выражении (F (G X)) список (G X) этим значением, то последующее вычисление функции F даст нужный результат.  
  
6. Принципы фон Неймана

**Принцип однородности памяти** -  Распознать их можно только по способу использования; то есть одно и то же значение в ячейке памяти может использоваться и как данные, и как команда, и как адрес в зависимости лишь от способа обращения к нему. Это позволяет производить над командами те же операции, что и над числами, и, соответственно, открывает ряд возможностей.  Более полезным является другое следствие принципа однородности, когда команды одной программы могут быть получены как результат исполнения другой программы. Эта возможность лежит в основе трансляции — перевода текста программы с языка высокого уровня на язык конкретной вычислительной машины.

**Принцип адресности -** Структурно основная память состоит из пронумерованных ячеек, причём процессору в произвольный момент доступна любая ячейка. Двоичные коды команд и данных разделяются на единицы информации, называемые словами, и хранятся в ячейках памяти, а для доступа к ним используются номера соответствующих ячеек — адреса

7. Структура ЭВМ?? - **?????**

8. внутреннее представление списка - **?????**

9. как система трактует элемент списка?? - **?????**

10. Как воспринимается символ ’, его назначение

Вызов **(quote  *x*)** всегда возвращает  *x*. Вычисление (оценка) значения *object* не производится, а поэтому он может представлять собой произвольный Лисп-объект, что позволяет ввести в программу на Лиспе константу, содержащую любую конструкцию.

Например:

(setq a 43)  
(list a (cons a 3))       ; (43 (43 . 3))

Поскольку формы **quote** используются очень часто, и набивать их название постоянно просто нерационально, для них было разработано ставшее общепринятым сокращение: любая форма  *f*, перед которой следует знак апострофа,  интерпретируется как аргумент функции **(quote  )**, которая как бы окружает  *f*, образуя конструкцию **(quote  f)**.

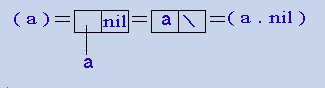
Например:

(setq x '(43))

интерпретируется модулем чтения **read** как, (setq x (quote (43)))

11. Как представляются списки в ОП?

Список из одного атома, представляется, как



NIL указывает на конец списка.

Вместо nil пишут - \.

Список получается как операция (cons 'a nil)

Любой список можно представить в точечной нотации.

(a) <=> (a.nil)

(a b c) <=> (a.(b.(c.nil)))

12. как выполняются функции car и cdr

car от списка - это первый элемент в списке. То есть car от списка (a b c d) будет а. || cdr от списка - это оставшаяся часть списка хвост, то есть, функция cdr возвращает часть списка, которая следует за первым элементом. cdr возвращает оставшуюся часть списка, то есть (b c d).