ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Лекция 2

ЭВОЛЮЦИЯ АРХИТЕКТУРЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

- Эра универсальных ЭВМ
- Эра персональных компьютеров
- Сетевая эра

ЭРА УНИВЕРСАЛЬНЫХ ЭВМ

- **Пакетные среды.** Самая ранняя и простая операционная среда полностью состояла из внешних файлов с данными. Такая операционная среда называется *средой пакетной обработки* (batch-processing). 80-колонная перфокарта или карта Холлерита была неотъемлемой частью компьютеров шестидесятых годов.
- В языках, разработанных для пакетной среды, файлы обычно являются основой для большинства структур ввода-вывода.
- При разработке этих языков возможностям обнаружения и обработки ошибок и исключительных ситуаций непосредственно в программе придавалось большое значение. Программа должна была обрабатывать наиболее вероятные ошибки и продолжать свое выполнение без каких-либо остановок.

ЭРА УНИВЕРСАЛЬНЫХ ЭВМ

- **Интерактивные среды.** В начале 70х гг., ближе к концу эпохи универсальных ЭВМ, появилось интерактивное программирование в связи с появлением компьютеров с возможностью ***разделения времени***. При использовании интерактивной среды пользователь общается с программой во время ее выполнения посредством дисплея, на который выводятся выходные данные, и клавиатуры или мыши, позволяющих вводить информацию.
- Операции интерактивного ввода-вывода существенно отличаются от стандартных операций с файлами и это затрудняет адаптацию подобных языков программирования к интерактивным средам (новые языки C и впоследствии C++).
- Также в интерактивных средах используется совершенно другой подход к обработке ошибок. Если пользователь вводит неправильные входные данные с клавиатуры, программа может вывести сообщение об ошибке и предложить произвести исправления. Средства языка для обработки ошибок внутри программы (например, возможность пропустить ошибку и выполнять программу дальше) становятся менее существенными.

ЭРА ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

- **Персональные компьютеры.** Семидесятые годы могут быть названы *эрой миникомпьютеров*.
- В 1978 году компания Apple выпустила компьютер Apple II, первый по-настоящему коммерческий персональный компьютер.
- Состояние дел изменилось в 1981 году. Фирма IBM выпустила свой персональный компьютер, а фирма Lotus разработала свое приложение Lotus 1–2–3, основанное на программе обработки электронных таблиц Visi-Calc.
- Началом современной эры персональных компьютеров можно считать январь 1984 года, когда была показана реклама компьютера Macintosh фирмы Apple, характеризовавшегося оконным графическим пользовательским интерфейсом с мышью для ввода данных.

ЭРА ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

- С появлением персональных компьютеров вновь изменилась роль языка. Во многих прикладных областях производительность перестала быть основным требованием. Благодаря достаточно низким ценам на компьютеры отпала необходимость в разделении времени. Задачей первостепенной важности стала разработка языков с хорошей интерактивной графикой.

- Естественной моделью для данной среды является объектно-ориентированное программирование. Использование языков Java и C++ с их иерархией классов облегчает взаимодействие с пакетами, разработанными сторонними производителями.

ЭРА ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

- **Среда встроенных систем.** Встроенные компьютеры являются боковой ветвью развития персональных компьютеров.
- ***Встроенной компьютерной системой*** называется система, которая управляет частью более крупной системы, такой как промышленный завод, станок, автомобиль, или даже тостер.
- Программы, написанные для встроенных систем, должны обращаться к нестандартным устройствам ввода-вывода через специальные процедуры, учитывающие все особенности конкретного устройства.
- Во встроенных системах особенно важна обработка ошибок. Обычно каждую программу составляют так, чтобы она могла самостоятельно обработать любую ошибку и принять меры для восстановления и продолжения своей работы.

ЭРА ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

- Встроенные системы, как правило, работают в *режиме реального времени*, то есть большая система, в которую интегрирована компьютерная система, требует от нее ответов на запросы и выдачу выходного сигнала в течение строго определенных интервалов времени.

- Наконец, встроенная компьютерная система обычно является *распределенной системой*, состоящей из нескольких компьютеров. Программа, работающая в таких распределенных системах, обычно состоит из ряда одновременно выполняемых задач, каждая из которых управляет одной из частей системы или наблюдает за ней.

СЕТЕВАЯ ЭРА

- **Распределенная обработка данных.** Для использования в крупных организациях были разработаны локальные вычислительные сети (ЛВС) с архитектурой *клиент — сервер*, использующие линии телекоммуникаций для связи между компьютерами

- *Программа-сервер* должна обеспечивать доступ к информации, а множественные *клиентские программы* могут запрашивать сервер для получения этой информации.

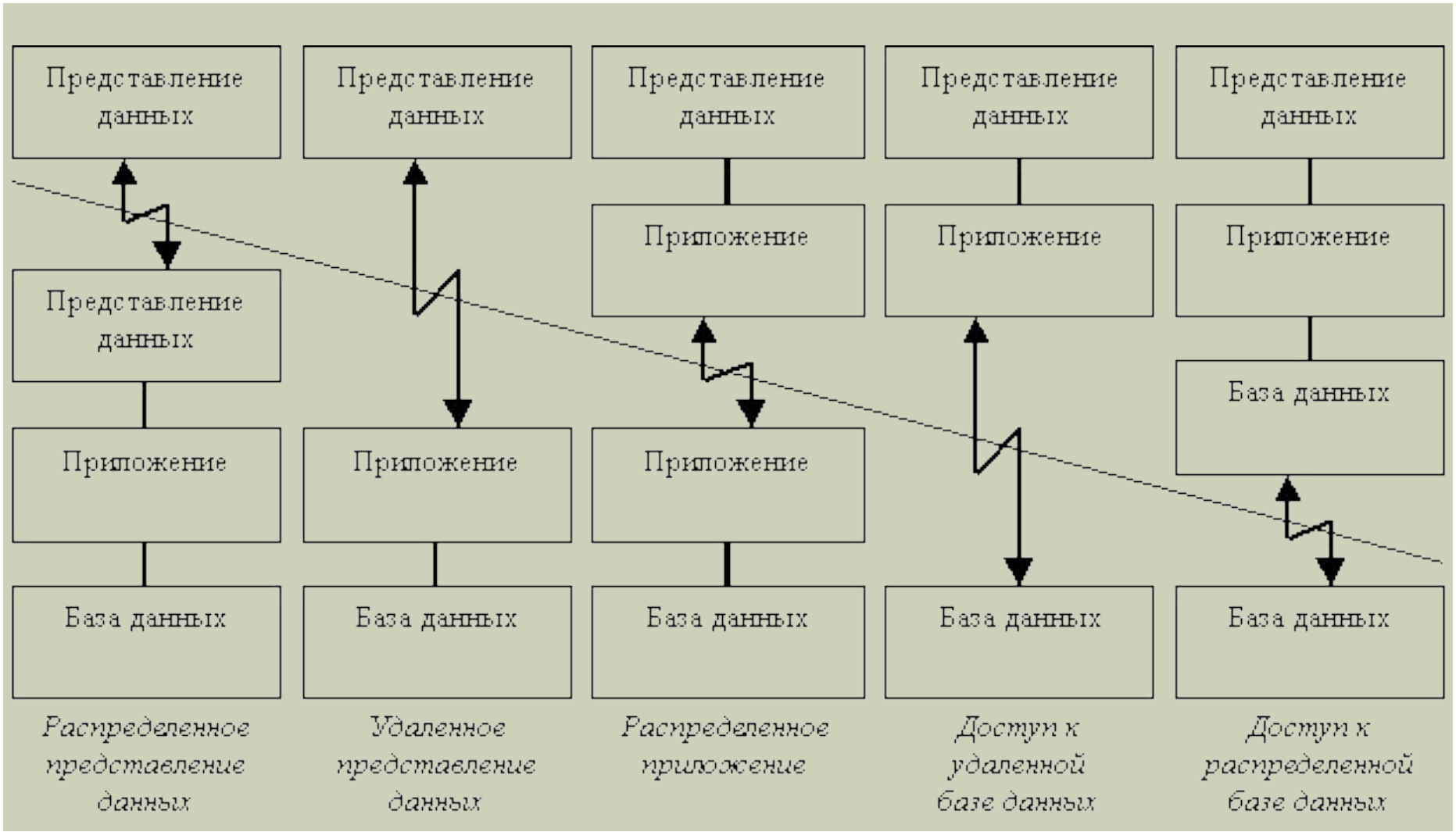
СЕТЕВАЯ ЭРА

- **Интернет.** В середине девяностых годов наблюдается преобразование распределенных ЛВС в международную глобальную сеть Интернет.
- Для получения информации использовались протоколы telnet, протокол передачи файлов FTP (file transfer protocol) и простой протокол передачи сообщений (SMTP — Simple Mail Transfer Protocol).
- В конце восьмидесятых Бернес-Ли (Bernes-Lee) разработал концепцию *гиперссылок* в рамках языка **HTML** (HyperText Markup Language) как способа навигации в Интернете. После создания в 1993 году Web-браузера Mosaic и добавления к Интернет-технологиям протокола передачи гипертекстов HTTP (HyperText Transfer Protocol) произошло «открытие» Интернета для широких слоев населения.

СЕТЕВАЯ ЭРА

- Вычисления снова стали централизованными, но существенно иным образом, нежели в раннюю эру универсальных компьютеров. По всему миру создаются крупные серверы информационных архивов. Для получения информации пользователи подключаются к этим серверам через Интернет, а для ее обработки (например, для создания отчета) используют локальные клиентские машины.
- Изначально Web-страницы были статическими. Однако для развития электронной коммерции информация должна передаваться в обоих направлениях между клиентской машиной и сервером, поэтому Web-страницы должны были стать более активными. Подобные возможности обеспечиваются такими языками программирования как **Perl** и **Java**.
- Использование WWW поставило перед языками такие проблемы, которые не были очевидны в предыдущие две эры. Одна из них — безопасность.
- Еще одна важная проблема — производительность. Чтобы разгрузить сервер за счет клиентской машины, он должен переслать клиенту небольшую исполняемую программу. Проблема состоит в том, что сервер не знает, каким компьютером является клиентская машина, поэтому не ясно, какого вида должна быть исполняемая программа.

СЕТЕВАЯ ЭРА:
МОДЕЛИ ВЗАИМОДЕЙСТВИЯ КЛИЕНТ-СЕРВЕР



ЯЗЫКИ ПРОГРАММИРОВАНИЯ В ПРИКЛАДНЫХ ОБЛАСТЯХ

Период	Приложение	Основные языки	Другие языки
1960–1970	Бизнес	COBOL	Assembler
	Наука	FORTRAN	ALGOL, BASIC, APL
	Система	Assembler	JOVIAL, FORTH
	ИИ	LISP	SNOBOL
Сегодня	Бизнес	COBOL, C++, Java, табличные процессоры	C, PL/I, 4GL
	Наука	FORTRAN, C, C++	BASIC, Pascal
	Система	C, C++	Ada, BASIC, Modula
	ИИ	LISP, Prolog	
	Издательство	TeX, Postscript, текстовые процессоры	
	Процессы	язык shell системы UNIX, TCL, PERL, JavaScript	AWK, sed, Marvel
	Новые парадигмы	ML, Smalltalk	Eifell

ПРОБЛЕМЫ РАЗРАБОТКИ ЯЗЫКА ПРОГРАММИРОВАНИЯ

- На принципы конструирования новых языков влияют следующие факторы:

1. Возможности компьютеров;
2. Области применения;
3. Методы программирования;
4. Методы реализации;
5. Теоретические исследования;
6. Стандартизация.

ПРОБЛЕМЫ РАЗРАБОТКИ ЯЗЫКА ПРОГРАММИРОВАНИЯ

- Свойства хорошего языка:

1. Ясность, простота и единообразие понятий;
2. Ортогональность;
3. Естественность для приложений;
4. Поддержка абстракций;
5. Удобство верификации программы;
6. Среда программирования;
7. Переносимость программ;
8. Стоимость использования.

ПРОБЛЕМЫ РАЗРАБОТКИ ЯЗЫКА ПРОГРАММИРОВАНИЯ

- Стоимость использования языка:

a) **Стоимость выполнения.** В прошлом самая важная стоимость. Специальные исследования оптимизации компилятора, использования регистров и т. п., особенно для больших программ, выполняемых достаточно часто. С настольными машинами это стало не так важно: большая скорость выполнения (несколько миллионов операций в секунду), больше простаивают.

b) **Стоимость трансляции.** В ЯП, используемых для обучения, — это основной критерий. До выполнения студент много раз транслирует, исправляя синтаксические и другие ошибки.

c) **Стоимость создания, тестирования и использования.** Одна из наиболее важнейших характеристик стоимости, наряду с выполнением и компилированием.

d) **Стоимость сопровождения программы.** Это основной период из жизненного цикла программы. Чем легче модифицировать, изменять, корректировать — тем лучше.

АЛГОРИТМЫ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ

- *Согласно гипотезе лингвистической относительности, иначе называемой «гипотеза Сепира-Уорфа», люди, говорящие на разных языках, по-разному воспринимают мир и по-разному мыслят.*

- **Алгоритм** - точное предписание исполнителю совершить определенную последовательность действий для достижения поставленной цели за конечное число шагов.

- Несмотря на изобилие способов представления алгоритмов, программисты обычно используют для создания алгоритма тот же язык программирования, на котором пишут программу, и придумывают алгоритм, исходя из возможностей этого языка.

ЗАДАЧА

- Найти, сколькими различными способами можно расставить N предметов (N может находиться в диапазоне от 13 до 20), причем предмет номер 0 должен находиться на 10-м месте справа.

РЕШЕНИЕ

- **Решение 1:** создаем полный список всех возможных перестановок, просматриваем его и считаем только те, которые удовлетворяют условию.

- Проблема в том, что, во-первых, при N=20 этот алгоритм потребует примерно два миллиона терабайт памяти, а во-вторых, будет выполняться более 2000 лет, хотя алгоритм этот формально правильный.

РЕШЕНИЕ

- **Решение 2:** Посчитать (N-1)!, при N=20 равное 121645100408832000.

- Проблема в том, что это число не влезает в 32 разряда (но влезает в 64). Следовательно, решение задачи сведется либо к созданию подпрограмм для работы с 64-битными целыми числами, либо к к созданию библиотеки для работы с числами произвольной разрядности. Однако, в некоторых языках возможность работы с такими числами входит в стандартные спецификации, так что задача сводится к написанию подпрограммы вычисления факториала числа.

АЛГОРИТМЫ И ЯЗЫКИ ПРОГРАММИРОВАНИЯ

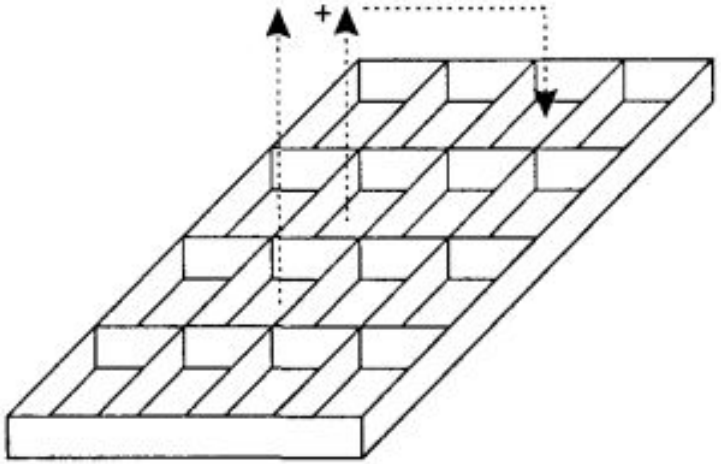
- Такая двойственность алгоритма (с одной стороны независимость от языка, а с другой - тесная связь с языком) объясняется достаточно просто. *Многие языки программирования схожи между собой.*

- Соответственно, алгоритмы, придуманные для одного языка, легко (или не очень легко) переносятся на другой.

ПАРАДИГМЫ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

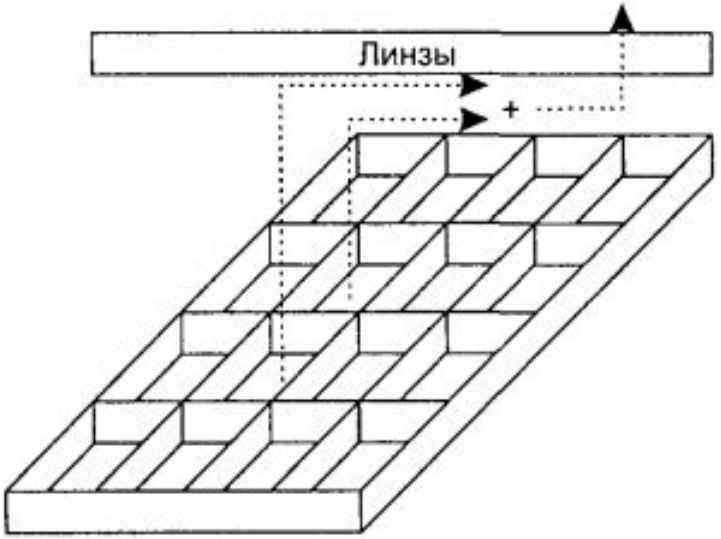
- **Парадигма ЯП** — это собрание основополагающих принципов, которые служат методической основой конкретных технологий и инструментальных средств программирования.

1. Императивная;
2. Аппликативная;
3. Основанная на системе правил;
4. Объектно-ориентированная.



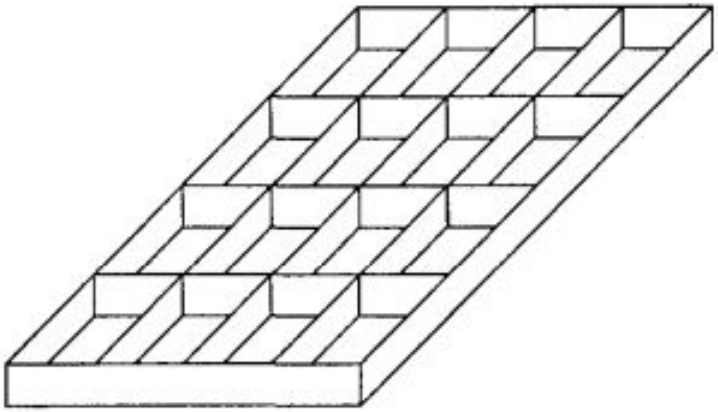
Императивные языки —
память состоит из коробочек

а



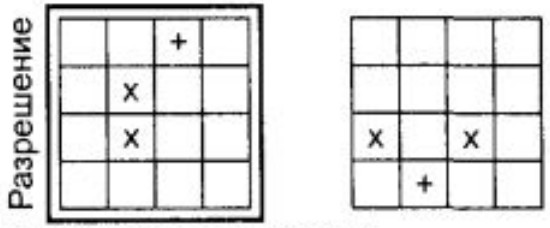
Аппликативные языки —
определяют изменения данных
в процессе извлечения их из памяти

б

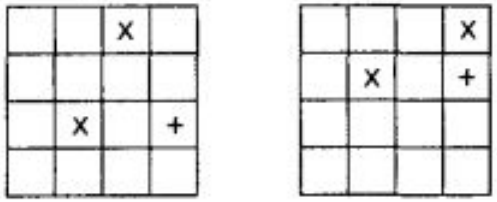


Языки, основанные на системе правил, —
используют фильтры для разрешения
изменения состояния

в



Такая же операция, что и в *а*



Каждая сетка представляет возможный фильтр,
определяя операции и операнды

г

ИМПЕРАТИВНАЯ МОДЕЛЬ

- *Императивные* или *процедурные языки* — это управляемые командами или операторно-ориентированные языки программирования. Основной концепцией является состояние машины — множество всех значений всех ячеек памяти компьютера.

- Программа состоит из последовательности операторов, выполнение каждого из которых влечет за собой изменение значения в одной или нескольких ячейках памяти, то есть переход машины в новое состояние. В целом синтаксис такого языка имеет вид:

```
оператор1;
оператор2;
```

АППЛИКАТИВНАЯ МОДЕЛЬ

- Другим взглядом на вычисления, производимые с помощью языка программирования, является рассмотрение функции, которую выполняет программа, а не отслеживания изменяемых состояний машины во время выполнения программы, оператор за оператором. Языки, в которых акцентирован именно этот взгляд на вычисления, называются *аппликативными* или *функциональными*.

- Синтаксис такого языка, как правило, выглядит следующим образом:

```
функцияn(…функция2(функция1(данные))…)
```

МОДЕЛЬ, ОСНОВАННАЯ НА СИСТЕМЕ ПРАВИЛ

- Языки, основанные на системе правил, осуществляют проверку наличия необходимого разрешающего условия, и в случае его обнаружения выполняют соответствующее действие. Наиболее распространенный язык, основанный на системе правил, — Prolog. Он также называется *языком логического программирования*, поскольку базовые разрешающие условия относятся к классу выражений логики предикатов.
- Выполнение программы на подобном языке похоже на выполнение программы, написанной на императивном языке, за исключением того, что операторы выполняются не в той последовательности, в которой они определены в программе. Разрешающие условия определяют порядок выполнения. Синтаксис таких языков выглядит следующим образом:

```
разрешающее условие1 → действие1
разрешающее условие2 → действие2
…
разрешающее условиеn → действиеn
```

- (Иногда правила записываются в виде `действие if разрешающее условие`, в котором выполняемое действие записывается слева.)

ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ МОДЕЛЬ

- В этой модели строятся сложные объекты данных, а затем для операций над этими данными описывается ограниченный набор функций. Сложные объекты создаются как расширения более простых объектов и наследуют их свойства. Как мы покажем, на самом деле эта модель является попыткой объединить лучшие свойства других моделей. Благодаря возможности строить конкретные объекты данных, объектно-ориентированная программа приобретает эффективность императивного языка. Построение классов функций, которые используют ограниченный набор объектов данных, дает нам гибкость и надежность, свойственные аппликативному языку.
- Основные принципы:

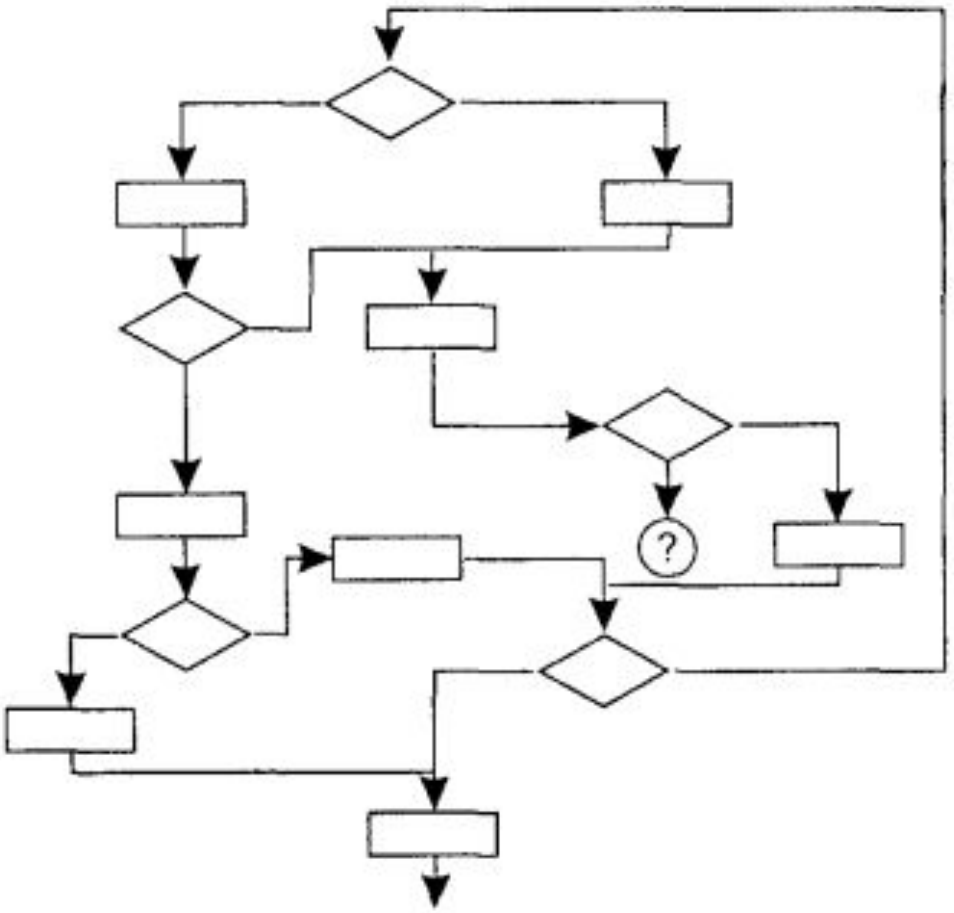
1. Инкапсуляция
2. Наследование
3. Полиморфизм

УНИВЕРСАЛЬНОСТЬ ВЫЧИСЛИТЕЛЬНОЙ МОДЕЛИ

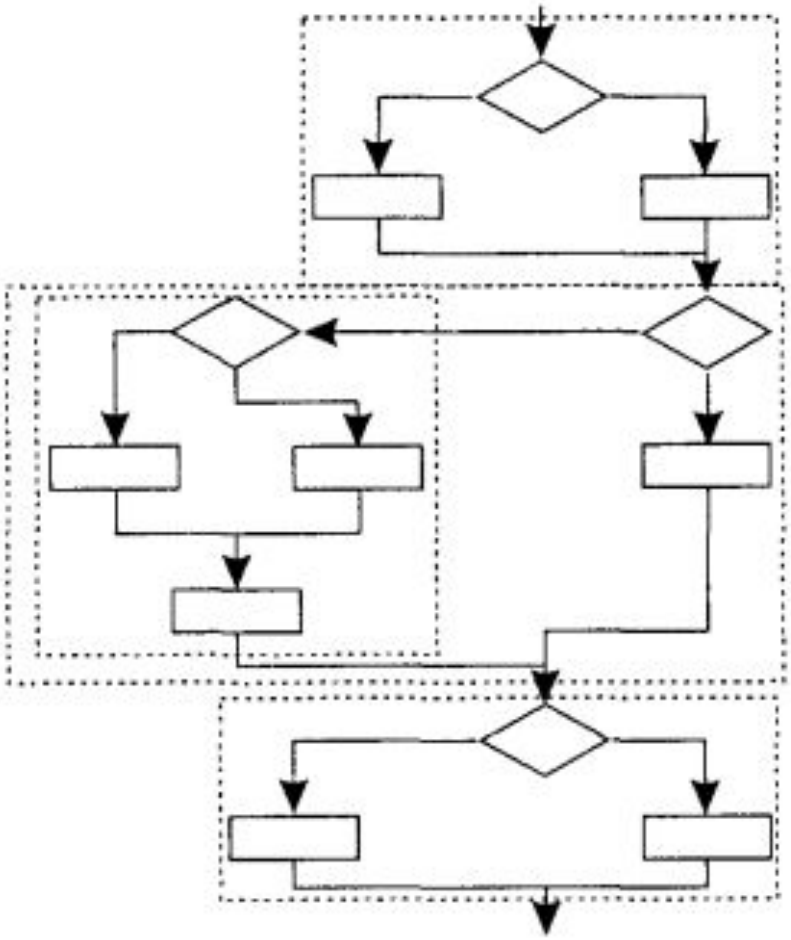
- **Парадигма программирования** — это комплекс концепций, принципов и абстракций, определяющих фундаментальный стиль программирования. Парадигма задается использованием определенных сущностей, например:

1. состояний программы и команд, изменяющих их (**императивное программирование**),
2. математических функций без состояний (**функциональное программирование**),
3. объектов и взаимодействий между ними (**объектно-ориентированное программирование**),
4. алгоритмов и контейнеров, оперирующих с типами данных, переданными как параметр (**обобщенное программирвоание**),
5. значений и операций, преобразующих значения (**программирование на уровне значений**), и т.д.

АППЛИКАТИВНЫЕ МЕТОДЫ В ИМПЕРАТИВНЫХ ЯЗЫКАХ



Программа-спагетти



Структурированная программа

ВЫВОДЫ

СЕМИНАР 12.09.2013

1. **Структура и принципы работы компьютера** [1] стр. 64-78
   1. Аппаратные средства компьютера
   2. Программно-аппаратный компьютер
   3. Трансляторы и виртуальные архитектуры
2. **Виртуальные компьютеры и время связывания** [1] стр. 78-87
   1. Виртуальные компьютеры и реализация языка
   2. Иерархия виртуальных компьютеров
   3. Связывание и время присваивания
3. **Обзор языка Java** [1] стр. 87-89
4. **Обзор языка C** [1] стр.58-61