Тема. Парадигмы и языки программирования

1. Основные парадигмы программирования.
2. Языки программирования: основные понятия, критерии оценки, предпосылки развития.

ПАРАДИГМА

Парадигма – это система взглядов на явления окружающего мира и представлений о возможных взаимодействиях с ними.

Парадигма в программировании – система идей и понятий, определяющих фундаментальный стиль программирования.

Основные парадигмы программирования

Парадигма (стиль) программирования:

-отображает определенную модель вычислений, включая структуры данных и механизмы управления;

-каждая парадигма ориентирована на определенный класс прикладных задач, которые удобно решать средствами данной парадигмы.

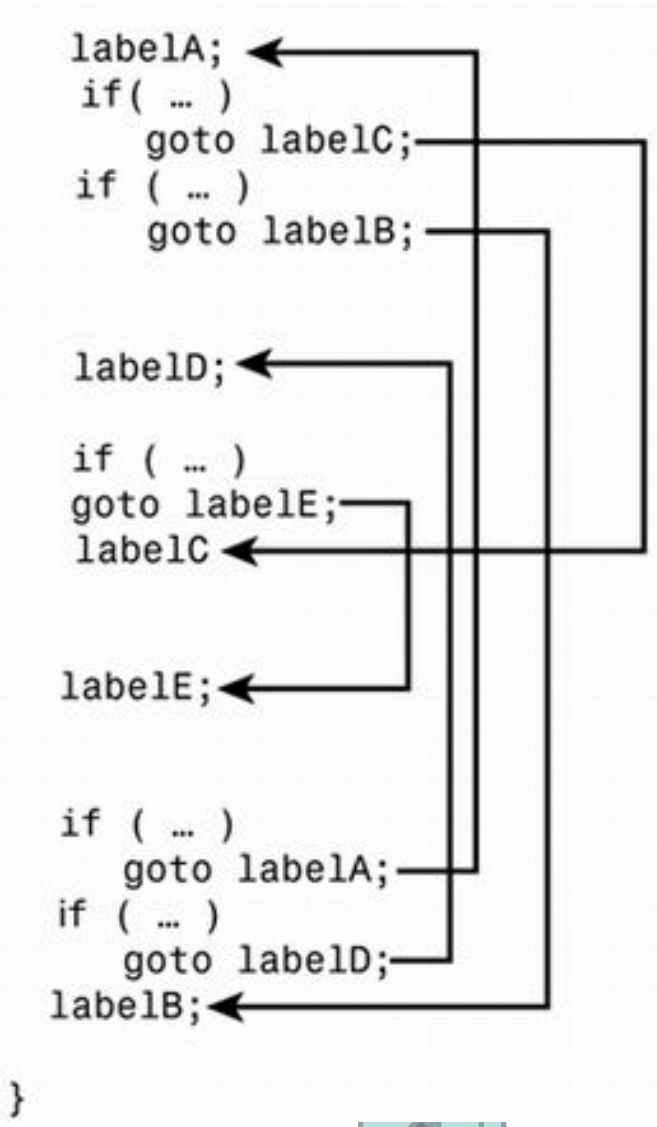
Большинство современных языков программирования обычно включают средства и приемы программирования различных парадигм.

Основные парадигмы программирования

Основные парадигмы программирования

1. Императивное программирование:

- «Код управляет данными»
- Программа = последовательность действий, связанных условными и безусловными переходами
- массовое использование goto
- «спагетти» («кенгуру») - код



Основные парадигмы программирования

Структурное программирование:

- программа представляется в виде иерархической структуры блоков
- доказано, что любая программа может быть построена из 3-х видов управляющих структур
- каждая логическая законченная группа инструкций должна быть оформлена в виде блоков (begin…end, {…})
- управляющие конструкции и блоки могут быть вложенными, в любой из них должен быть один вход и один выход

Основные парадигмы программирования

Процедурное программирование:

- представляет собой развитие структурного программирования
- повторяющиеся фрагменты программы оформляются в виде подпрограмм:

❖ процедуры – не имеют возвращаемого значения, могут изменять параметры;

❖ функции – возвращают единственное значение, могут изменять параметры

Основные парадигмы программирования

Модульное программирование:

- структуры данных и подпрограммы для работы с ними объединяются в модули:

❖открытая часть модуля (interface) – видна другим модулям;

❖закрытая часть модуля (implementation) – видна только внутри модуля

Основные парадигмы программирования

2. Декларативное программирование:

- «данные управляют программой»
- Программа заявляет (декларирует), что должно быть достигнуто в качестве цели
- Алгоритм решения не задается, от значения данных, поступающих на вход программы, зависит направление расчетов

```
SELECT имя_столбца FROM имя_таблицы ORDER BY имя_столбца_сортировки
```

Основные парадигмы программирования

Функциональное программирование:

- основано на математическом понятии функции
- программа состоит из совокупности определений функций, которые в свою очередь представляют собой вызовы других функций и предложений, управляющих последовательностью вызовов

Основные парадигмы программирования

Функциональное программирование:

- каждая функция возвращает некоторое значение в вызвавшую его функцию, вычисление которой после этого продолжается; этот процесс повторяется до тех пор, пока не будет достигнут результат.

```
// Evaluate a number's factorial:

(defun factorial (n)
  (if (= n 0) 1
      (* n (factorial (- n 1)))))


// Common Lisp's loop macro

(defun factorial (n)
  (loop for i from 1 to n
        for fac = 1 then (* fac i)
        finally (return fac)))
```

```
// takes less stack space

(defun factorial (n &optional (acc 1))
   (if (= n 0) acc
       (factorial (- n 1) (* acc n))))


// The following function reverses a list.

(defun -reverse (list)
  (let ((return-value '()))
    (dolist (e list) (push e return-value))
    return-value))
```

Основные парадигмы программирования

Логическое программирование:

Метод программирования, предназначенный для решения задач искусственного интеллекта, в соответствии с которым программа описывает логическую структуру решения задачи, указывая преимущественно, что «нужно сделать», не вдаваясь в детали «как это делается». Практическим воплощением метода является язык Пролог.

Основные парадигмы программирования

3. Объектно-ориентированное программирование:

- Первичными считаются объекты (данные), которые взаимодействуют друг с другом посредством механизма передачи сообщений.
- Ключевые понятия:

❖Инкапсуляция

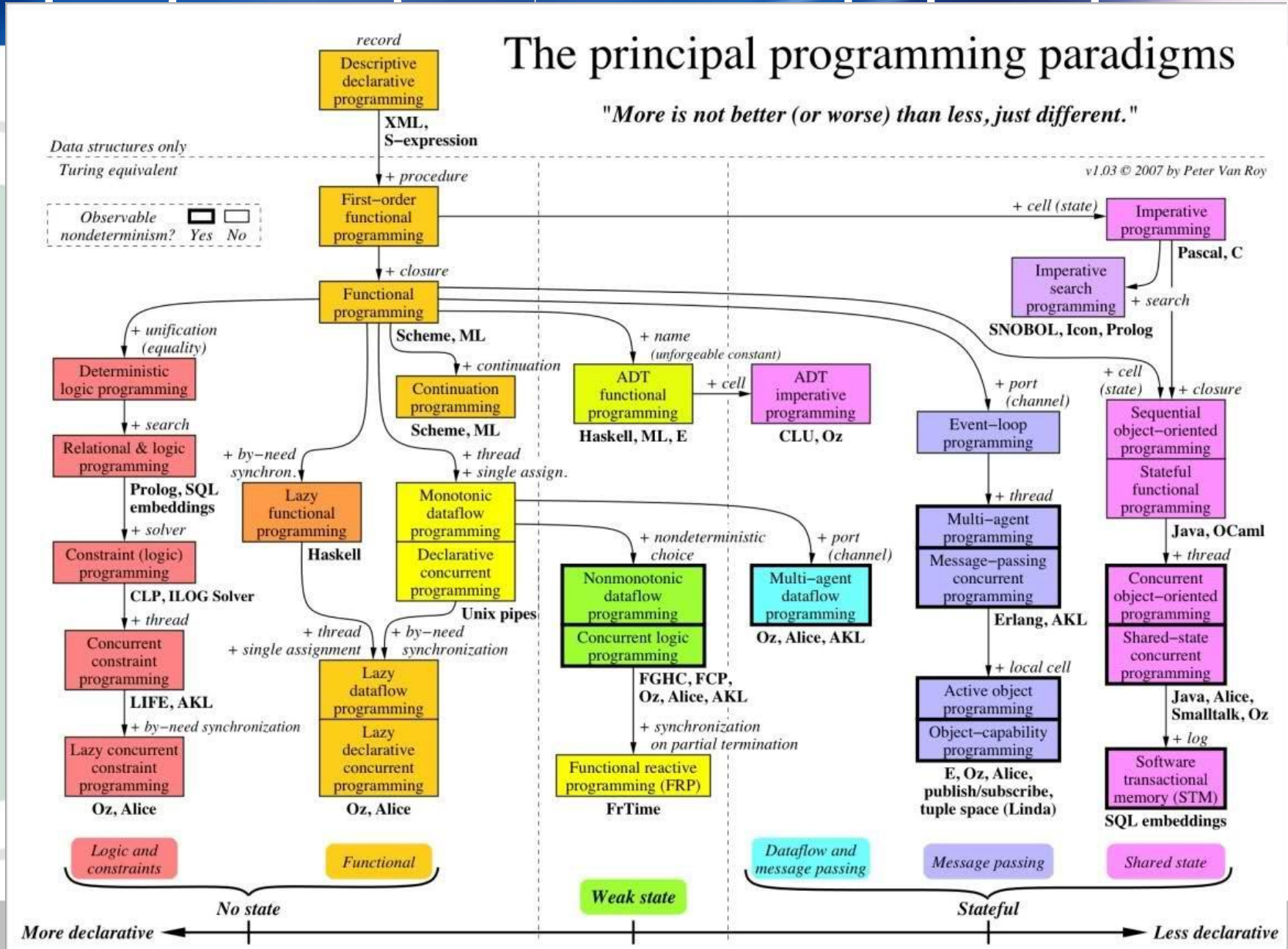
❖Наследование

❖Полиморфизм

Основные парадигмы программирования

- Инкапсуляция – структуры данных и подпрограммы для работы с этими данными объединены в «классы»
- Наследование – возможность описать новый класс на основе уже описанного «родительского» класса (с различными спецификаторами доступами)
- Полиморфизм – определение одноименных методов для различных «родственных» классов и способность во время выполнения выбирать и выполнять тот метод, который соответствует полученному объекту

Пример классификации языков программирования



Языки программирования

Содержание:

- Структура языка программирования
- Составляющие ЯП высокого уровня (алфавит, синтаксис, семантика)
- Понятие о формальных грамматиках и синтаксических диаграммах
- Характеристики ЯП и их влияние на критерии оценки программного обеспечения
- Развитие языков программирования

Структура языка программирования

Составляющие ЯП

Алфавит - основные символы языка, используемые при написании программ, как правило, включает в себя:

- строчные и прописные буквы латинского алфавита
- цифры
- знаки операций: + - * / = < : @ & |
- символы подчеркивания _ и пробела
- ограничители и разделители: . , ‘ ( ) { } [ ]
- специальные символы: ^ # $ и др.
- комбинации специальных символов, которые нельзя разделять пробелами, если они используются как знаки операций: “:=“, “..”, “<>”, “<=”, “>=”, “{ }”

Составляющие ЯП

Лексика:

- может рассматриваться как составляющая алфавита
- включает в себя:
  - совокупность правил образования цепочек символов (лексем), образующих идентификаторы (переменные и метки) и числа
  - операторы
  - операции
  - зарезервированные (запрещенные, ключевые) слова ЯП, предназначенные для обозначения операторов, встроенных функций

Составляющие ЯП

Лексика:

На этапе компиляции компилятор заменяет распознанные лексические конструкции предопределенными внутренними кодами и в дальнейшем рассматривает их как отдельные внутренние символы (лексемы)

Составляющие ЯП

Синтаксис:

- Совокупность правил образования языковых конструкций (предложений языка программирования) – блоков, процедур, составных операторов, условных операторов, операторов цикла и пр.
- Принцип вложенности правил построения конструкций (элемент синтаксиса языка в своем определении прямо или косвенно в одной из его частей содержит сам себя)

Составляющие ЯП

Синтаксис:

- Нарушение синтаксиса проверяется компилятором на этапе компиляции программы
- Примеры синтаксических ошибок:

❖ неверное описание названия функции при ее вызове;

❖ неверное количество аргументов;

❖ неверный тип переданных аргументов;

❖ неверный тип возвращаемого значения.

Составляющие ЯП

Семантика:

- смысловое содержание конструкций, предложений языка;
- семантический анализ – проверка смысловой правильности конструкции;
- семантические ошибки возникают при недопустимом использовании операций, массивов, функций, операторов и пр.

Составляющие ЯП

Способы описания синтаксиса ЯП:

- Для описания синтаксиса используется метаязык (надъязык), предназначенный для описания других языков.
- Наиболее распространенные метаязыки:

❖ форма Бэкуса-Наура (язык БНФ) – 60-е года XX века, John Backus, Peter Naur);

❖ синтаксические диаграммы (Вирт)

Основные обозначения и правила БНФ

Обозначения и правила	Правило трактования	Пример	Трактование
имя, записанное слитно русскими или латинскими буквами	конструкция языка	“оператор_цикла”	
квадратные скобки [элемент]	элемент языка внутри скобок может повторяться0 или 1 раз	“AB[C]”	в конструкции языка может присутствовать или AB или ABC
фигурные скобки [элемент]	элемент языка может повторяться 0 или много раз	“AB{C}”	в конструкции языка может присутствовать или AB или ABC
символ “|”	“или”	“AB”|”C”|”ff”	в конструкции языка может присутствовать или AB или C или ff

Составляющие ЯП

Формула Бэкуса-Наура:

- Примеры описания элементов языка С с помощью БНФ

Идентификатор в языке С

```
Идентификатор =
    (Буква | "_") { Буква | Цифра | "_" }
Буква =
"A" | "B" | . . . | "Y" | "Z" | "a" | "b" | . . . | "y" | "z"
Цифра = "0" | "1" | . . . | "9"
```

Составляющие ЯП

Синтаксические диаграммы (Вирта):

- Каждое определение представляет собой ориентированный граф с одним входом и одним выходом, в котором:

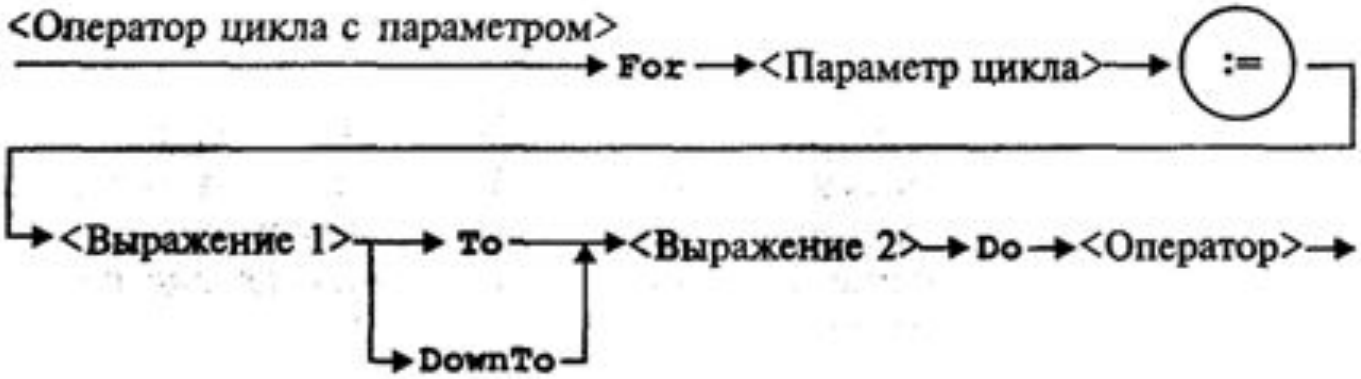
-вершины – это:

- ❖или терминальные символы;
- ❖или определяемые синтаксические понятия;
- ❖или разветвления/соединения на последовательность расположения элементов синтаксической конструкции

Составляющие ЯП

Синтаксические диаграммы:

-дуги (стрелки) указывают на последовательность расположения элементов синтаксической конструкции



Здесь <параметр цикла>::=<имя простой переменной порядкового типа>

Характеристики и критерии оценки и сравнения языков программирования

Некоторые характеристики ЯП:

- простота;
- управляющие структуры;
- типы и структуры данных;
- синтаксическая структура;
- поддержка абстракции;
- выразительность;
- проверка типов;
- обработки исключительных ситуаций

Критерии оценки программного обеспечения:

- читабельность программного кода;
- трудоемкость разработки программного кода;
- надежность программного обеспечения;
- «стоимость» языка программирования

Характеристики языков программирования

Простота:

-количеством элементарных конструкций, операторов;

-количество возможных способов совершения одного действия;

```
count = count + 1;
count += 1;
count++;
++ count;
```

-перегрузка операторов.

Характеристики языков программирования

Управляющие операторы:

-поддержка оператора безусловного оператора перехода goto;

-наличие программных конструкций, позволяющих легко обойтись без goto

Типы и структуры данных:

-поддержка расширенного набора типов данных:

•логический тип, символьный тип;

•структуры, записи, перечисления, объединения, …

•проверка типов при выполнении операций

Характеристики языков программирования

Синтаксическая структура:

-ограничения на способ записи идентификаторов;

-специальные слова и их использование;

```
procedure (…)
…
begin
   while (…) begin
      if (…) begin
         …
      end
   end
end
```

Pascal

```
procedure (…)
…
begin
   while (…) begin
      if (…) begin
         …
      end if
   end while
end proc
```

Ada

-возможность зависимости значения оператора от контекста использования

Характеристики языков программирования

Поддержка абстракции:

-абстракция – возможность определить и использовать сложные структуры или операции, игнорируя сложные детали реализации;

-абстракция процесса (процедура);

-абстракция данных (пример: реализация двоичного дерева, хранящего целочисленные значения)

Влияние характеристик ЯП на ключевые критерии оценки ПО

Характеристика ЯП	Критерии оценки		
	Читабель-ность	Трудоемкость разработки	Надежность
Простота	*	*	*
Управляющие структуры	*	*	*
Типы и структуры данных	*	*	*
Синтаксическая структура	*	*	*
Поддержка абстракции		*	*
Выразительность		*	*
Проверка типов			*
Обработка исключительных ситуаций			*
Ограниченное совмещение имен			*

Развитие языков программирования

Основные движущие силы эволюции ЯП:

-создание более совершенных программ;

-повышение эффективности процесса производства ПО;

-появление новых, специализированных задач, прикладных областей;

-более полное использование новых возможностей ЭВМ;

-развитие пользовательских интерфейсов;

-широкий спектр аппаратных устройств (кроссплатформенность);

-увеличение продолжительности жизненного цикла программы в условиях совершенствования аппаратных платформ и системного ПО (ОС и т.п.)

Появление языков программирования

Наиболее популярные ЯП высокого уровня:

-1951г. – первый компилятор (Grace Hopper);

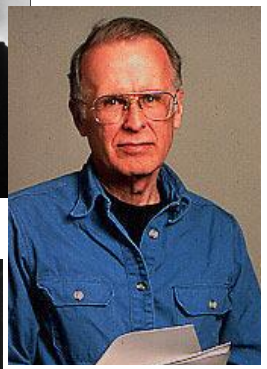
-1957г. – процедурный язык Фортран (FORmula TRANslator, John Backus);

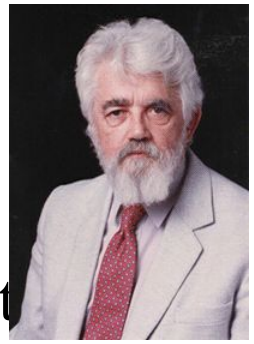
-1967г. – функциональный язык ЛИСП (John McCarthy);

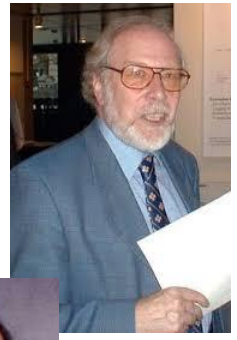
-1970г. – структурный язык Паскаль (Niklaus Wirt

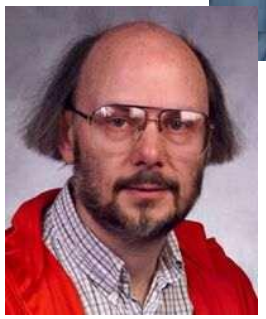
-1974г. – логический язык Пролог (Robert Kowalski, Maarten van Emden, Alain Colmerauer);

-1983г. – ОО язык C++ (Bjarne Stroustrup)









Появление языков программирования

Современные перспективные ЯП и их общие черты:

-сочетают особенности различных парадигм (“мультипарадигменные”);

-разработка/поддержка ИТ-гигантов (Apple, Facebook, Google, Microsoft);

-JIT-компиляция (Just-In-Time) – промежуточная между компиляцией и интерпретацией

Появление языков программирования

JIT-компиляция: программа компилируется на языке высокого уровня в некоторое промежуточное представление и это представление выполняется на виртуальной машине, которая своя для разной архитектуры. Благодаря этому код, написанный на языках NET-платформы или Java-платформы может выполняться на любой ОС.