Парадигмы программирования

Идеи и понятия: как писать программу

Императивная

- Инструкции, изменяющие состояние программы
- Основной оператор - присваивание
- a=b+c;
- ADD RAX, RBX

Процедурная

- Последовательное выполнение операторов с целью преобразования исходного состояния памяти
- P+=7;

Процедурная

- Сборка последовательных операторов в подпрограммы средствами языка

Структурная

- Последовательное исполнение
- Однократное выполнение операций в зависимости от условия
- addpl r0, r1, r2
- Прибавить r0=r1+r2 только если флаг знака не установлен (ARM)

Структурная

- Циклическое исполнение
- While (x<y) z++;

Структурная

- Вызов подпрограмм
- Scanf(“%d”,&x);
- Разработка “Сверху вниз”

Теорема Бома-Якопини (Бёма-Якопини)

- Любой исполняемый алгоритм может быть преобразован к структкрированному виду, то есть состоять из последовательного исполнения, ветвления и циклов.
- GOTO не нужно
- Подпрограммы не нужны

Модульная парадигма

- Программа разбивается на отдельные модули
- Известна функциональность модуля и его связи
- Структуры данных
- Библиотеки функций
- Классы
- Сервисы

Аспектно-ориентипрованное программирование

- Программа разделена на модули
- Функциональность, разделённая между модулями – сквозная
- Try – catch
- Логирование
- Проверка условий (в т. ч. прав доступа)

Объектно-ориентированное

- Абстракция (материальная точка)
- Инкапсуляция (no user serviceable parts inside)
- Наследование (квадрат - прямоугольник)
- Полиморфизм (сумма целых-целое, сумма действительных -действительное)
-

Агентно-ориентированное программирование

- Объект – сообщения только в ответ на сообщения
- Актор содержит данные и процедуры и может порождать сообщения
- Агент воспринимает динамику среды, изменяет её, выводит заключения о среде.

Компонентно-ориентированное программирование

- Компонент – независимый (в том числе от языка) модуль программного кода, предназначенный для повторного использования и развёртывания
- CORBA

Прототипно-ориентированное

- Наследуется не только структура, но и данные
- var foo = {name: "foo", one: 1, two: 2};
- var bar = {two: "two", three: 3};
- bar.one // Равно 1
- bar.three // Равно 3
- bar.two; // Равняется "two"

Обобщённое программирование

- Один алгоритм применяется ко многим типам данных

Декларативное

- Какие данные?
- Пример

Декларативное

- Какие данные?
- Пример

Функциональное программирование

- Чистые функции зависят только от аргументов и возвращают только результат
- Функция высшего порядка может иметь функции как аргументы и возвразщать функции

Логическое программирование

- Автоматическое доказательство теорем без перебора вариантов

Программирование потоком данных

- Нет произвольного доступа к данным
- AWK
- Сетевое оборудование
- Шаблон ->Действие

Метапрограммирование

- Автоматическая генерация кода
- Самомодифицирующийся код

Параллельное программирование

- Потоки исполняются одновременно на одной или нескольких машинах, конкурируют за общие ресурсы и взаимодействуют через сообщения

Событийно-ориентированное программирование

- Обработчики событий
- Векторы прерываний

Динамическое программирование

- Сведение задач к подзадачам

Классификация языков программирования

По близости к аппаратуре

- Машинно-независимые (javascript, C)
- Машинно-зависимые (ассемблер, байт-код)
- Языки описания аппаратуры (Verilog, VHDL)
-

По типам данных

- Статическая (тип постоянен)
- Динамическая (тип меняется)
- Явная (int x;)
- Неявная (x=6;)

По методу исполнения

- Компиляция в исполняемый код (C, fortran)
- Компиляция в промежуточный байт-код (Java, .NET)
- Интерпретация (bash, python)

По управлению памятью

- Объекты на стеке
- Указатели
- Ручное выделение памяти
- Сборка мусора

По применению

- Общего назначения
- Специального назначения

Как описать стандарт?

Request for comments - RFC

- Internet Draft
- Proposed Standart
- Draft Standart
- Internet Standart
- Historic Standart

Формы Бэкуса-Науэра

<определяемый символ> ::= <посл.1> | <посл.2> | . . . | <посл.n>