2. Overview of the four main programming paradigms

In this section we will characterize the four main programming paradigms, as identified in Section 1.2.

As the main contribution of this exposition, we attempt to trace the *basic discipline* and the *idea* behind each of the main programming paradigms.

With this introduction to the material, we will also be able to see how the functional programming paradigm corresponds to the other main programming paradigms.

2.1. Overview of the imperative paradigm

Contents Up Previous Next Slide Subject index Program index Exercise index

First **do this** and next **do that**

The 'first do this, next do that' is a short phrase which really in a nutshell describes the spirit of the imperative paradigm. The basic idea is the command, which has a measurable effect on the program state. The phrase also reflects that the order to the commands is important. 'First do that, then do this' would be different from 'first do this, then do that'.

In the itemized list below we describe the main properties of the imperative paradigm.

- Characteristics:
  - Discipline and idea
    - Digital hardware technology and the ideas of Von Neumann
  - Incremental *change of the program state* as a function of *time*.
  - Execution of computational steps in an order governed by *control structures*
    - We call the steps for *commands*

- Straightforward abstractions of the way a traditional Von Neumann computer works
- Similar to descriptions of everyday routines, such as food recipes and car repair
- Typical commands offered by imperative languages
  - Assignment, IO, procedure calls
- Language representatives
  - Fortran, Algol, Pascal, Basic, C
- The natural abstraction is the procedure
  - Abstracts one or more actions to a procedure, which can be called as a single command.
  - "Procedural programming"

We use several names for the computational steps in an imperative language. The word *statement* is often used with the special computer science meaning 'a elementary instruction in a source language'. The word *instruction* is another possibility; We prefer to devote this word the computational steps performed at the machine level. We will use the word 'command' for the imperatives in a high level imperative programming language.

A procedure abstracts one or more actions to a procedure, which can be activated as a single action.

2.2. Overview of the functional paradigm

Contents Up Previous Next Slide Subject index Program index Exercise index

We here introduce the functional paradigm at the same level as imperative programming was introduced in Section 2.1.

Functional programming is in many respects a simpler and more clean programming paradigm than the imperative one. The reason is that the paradigm originates from a purely mathematical discipline: the theory of functions. As described in Section 2.1, the imperative paradigm is rooted in the key technological ideas of the digital computer, which are more complicated, and less 'clean' than mathematical function theory.

Below we characterize the most important, overall properties of the functional programming paradigm. Needless to say, we will come back to most of them in the remaining chapters of this material.

Evaluate an expression and use the resulting value for something

- Characteristics:
  - Discipline and idea
    - Mathematics and the theory of functions
  - The values produced are *non-mutable*
    - Impossible to change any constituent of a composite value
    - As a remedy, it is possible to make a revised copy of composite value
  - Atemporal
    - Time only plays a minor role compared to the imperative paradigm
  - Applicative
    - All computations are done by applying (calling) functions
  - The natural abstraction is the function
    - Abstracts a single expression to a function which can be evaluated as an expression
  - Functions are first class values
    - Functions are full-fledged data just like numbers, lists, ...
  - Fits well with computations driven by needs
    - Opens a new world of possibilities

2.3. Overview of the logic paradigm

Contents Up Previous Next Slide Subject index Program index Exercise index

The logic paradigm is dramatically different from the other three main programming paradigms. The logic paradigm fits extremely well when applied in problem domains that deal with the extraction of knowledge from basic facts and relations. The logical paradigm seems less natural in the more general areas of computation.

Answer a question via search for a solution

Below we briefly characterize the main properties of the logic programming paradigm.

- Characteristics:
  - Discipline and idea
    - Automatic proofs within artificial intelligence
  - Based on axioms, inference rules, and queries.
  - Program execution becomes a systematic search in a set of facts, making use of a set of inference rules

2.4. Overview of the object-oriented paradigm

Contents Up Previous Next Slide Subject index Program index Exercise index

The object-oriented paradigm has gained great popularity in the recent decade. The primary and most direct reason is undoubtedly the strong support of encapsulation and the logical grouping of program aspects. These properties are very important when programs become larger and larger.

The underlying, and somewhat deeper reason to the success of the object-oriented paradigm is probably the conceptual anchoring of the paradigm. An object-oriented program is constructed with the outset in concepts, which are important in the problem domain of interest. In that way, all the necessary technicalities of programming come in second row.

Send messages between objects to simulate the temporal evolution of a set of real world phenomena

As for the other main programming paradigms, we will now describe the most important properties of object-oriented programming, seen as a school of thought in the area of computer programming.

- Characteristics:
  - Discipline and idea
    - The theory of concepts, and models of human interaction with real world phenomena
  - Data as well as operations are encapsulated in objects
  - Information hiding is used to protect internal properties of an object
  - Objects interact by means of message passing

    - A metaphor for applying an operation on an object
- In most object-oriented languages objects are grouped in classes
    - Objects in classes are similar enough to allow programming of the classes, as opposed to programming of the individual objects
    - Classes represent concepts whereas objects represent phenomena
- Classes are organized in inheritance hierarchies
    - Provides for class extension or specialization

This ends the overview of the four main programming paradigms. From now on the main focus will be functional programming in Scheme, with special emphasis on examples drawn from the domain of web program development.

Generated: Tuesday July 2, 2013, 09:27:19