



Our first paradigm

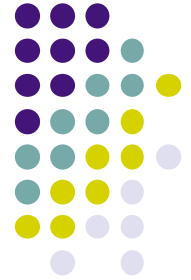
- Functional programming
 - It is one of the simplest paradigms
 - It is the foundation of all the other paradigms
 - It is a form of *declarative programming*
- Our approach to functional programming
 - It is our first introduction to programming **concepts**
 - It is our first introduction to a **kernel language**
 - We use it to explain **invariants and recursion**
 - We give examples using **integers, lists, and trees**
 - We present **higher-order programming**: the apotheosis
 - We give a **formal semantics** based on the kernel language

Declarative programming: the long-term view



- Declarative programming is **a vision for the future**
 - Just say **what** result you want (give properties of the result)
 - Let the computer figure out **how** to get there
 - **Declarative versus imperative: *properties versus commands***
- How do we make this vision real
 - Programming gets more support from the computer
 - With same programming effort, we can do more
- The whole history of computing is a progression toward **more declarative**
 - And faster and cheaper (all three are connected)

Declarative programming: the short-term view



- Declarative programming is **the use of mathematics in programming** (such as **functions** and **relations**)
 - A computation calculates a function or a relation
 - Use the power of mathematics to simplify programming (such as confluency and referential transparency)
- Very common in practice
 - Functional languages: LISP, Scheme, ML, Haskell, OCaml, ...
 - Logic languages (relational): SQL, constraint programming, Prolog, ...
 - Combinations: XSL (formatting), XSLT (transforming), ...
- Also called “programming without state”
 - Variables and data structures can’t be updated
 - Testing and verification is much simplified
 - **Declarative versus imperative: *stateless* versus *stateful***

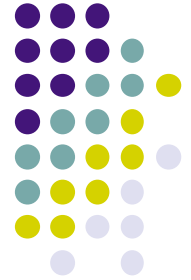
Key advantage of functional programming



- “A program that works today will work tomorrow”
 - Functions don’t change
 - All changes are in the arguments, not in the functions
- It is a programming style that should be encouraged in all languages
 - “Stateless server” for a client/server application
 - “Stateless component” for a service application
- Learning functional programming helps us think in this style
 - All programs written in the functional paradigm are ipso facto declarative: an excellent way to learn to think declaratively

“Cookies” on the Web

Now let's start programming...



- This completes the « philosophical » introduction of the course
- Now we will start programming in our first paradigm
 - Functional programming
- At the same time, we will introduce the Oz language and the Mozart system
 - Mozart's emacs interface, which we will use throughout the course