



UC Berkeley
Computer Science
Lecturer SOE
Dan Garcia

CS39N The Beauty and Joy of Computing

Lecture #6 : Programming Paradigms

2009-09-28



BRITISH PM APOLOGIZES TO TURING

In response to a 30,000+ signature petition, British PM Gordon Brown apologized for the way Britain treated Alan Turing (the "father" of computer science, WWII codebreaker) for being gay. *So on behalf of the British government, ... we're sorry, you deserved so much better.*

news.bbc.co.uk/2/hi/technology/8249792.stm
www.number10.gov.uk/Page20571

Programming Paradigms Lecture

- What are they?
 - Most are Hybrids!
- The Four Primary ones
 - Functional
 - Imperative
 - Object-Oriented
 - OOP Example: Sketchpad
 - Declarative
- Turing Completeness
- Summary



UC Berkeley CS39N "The Beauty and Joy of Computing" : Programming Paradigms (3)



en.wikipedia.org/wiki/Programming_paradigm

What are Programming Paradigms?

- "The concepts and abstractions used to represent the elements of a program (e.g., objects, functions, variables, constraints, etc.) and the steps that compose a computation (assignment, evaluation, continuations, data flows, etc.)."
- Or, a way to classify the style of programming.



UC Berkeley CS39N "The Beauty and Joy of Computing" : Programming Paradigms (3)



Most Languages Are Hybrids!

- This makes it hard to teach to students, because most languages have facets of several paradigms!
 - Scratch too!
- It's like giving someone a juice drink (with many fruit in it) and asking to taste just one fruit!



UC Berkeley CS39N "The Beauty and Joy of Computing" : Programming Paradigms (4)

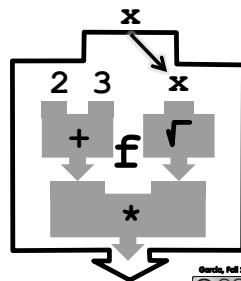


en.wikipedia.org/wiki/Functional_programming

Functional Programming (review)

- Computation is the evaluation of math functions
 - Plugging pipes together
 - Each pipe, or function, has exactly 1 output
 - Functions can be input!
- Features
 - No state
 - E.g., variable assignments
 - No mutation
 - E.g., changing variable values
 - No side effects
- Examples
 - Scheme, Scratch BYOB

$$f(x) = (2+3) * \sqrt{x}$$



UC Berkeley CS39N "The Beauty and Joy of Computing" : Programming Paradigms (3)



en.wikipedia.org/wiki/Imperative_programming

Imperative Programming

- AKA "Sequential" Programming
- Computation a series of steps
 - Assignment allowed
 - Setting variables
 - Mutation allowed
 - Changing variables
- Like following a recipe. E.g.,
- Procedure f(x)
 - ans = x
 - ans = $\sqrt{\text{ans}}$
 - ans = (2+3) * ans
 - return ans
- Examples: Pascal, C



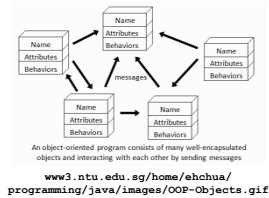
UC Berkeley CS39N "The Beauty and Joy of Computing" : Programming Paradigms (4)



en.wikipedia.org/wiki/Object-oriented_programming

Object-Oriented Programming (OOP)

- **Objects as data structures**
 - With **methods** you ask of them
 - These are the behaviors
 - With **local state**, to remember
 - These are the attributes
- **Classes & Instances**
 - Instance an example of class
 - E.g., Fluffy is instance of Dog
- **Inheritance saves code**
 - Hierarchical classes
 - E.g., pianist special case of musician, a special case of performer
- **Examples: Java, C++**



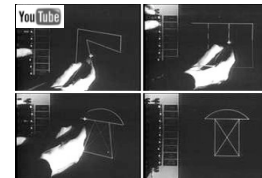
en.wikipedia.org/wiki/Sketchpad

OOP Example : SketchPad

- **Dr. Ivan Sutherland**
 - "Father of Computer Graphics"
 - 1988 Turing Award ("Nobel prize" for CS)
 - Wrote Sketchpad for his foundational 1963 thesis
- **The most impressive software ever written**
- **First...**
 - Object-oriented system
 - Graphical user interface
 - non-procedural language



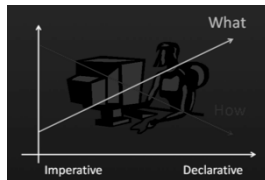
Spent the past few years doing research @ Berkeley in EECS dept!



en.wikipedia.org/wiki/Declarative_programming

Declarative Programming

- **Express what computation desired without specifying how it carries it out**
 - Often a series of assertions and queries
 - Feels like magic!
- **Sub-categories**
 - Logic
 - Constraint
 - We saw in Sketchpad!
- **Examples: Prolog**

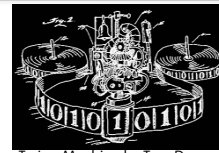


Anders Hejlsberg
"The Future of C#" @ PDC2008
channel9.msdn.com/pdc2008/TL16/

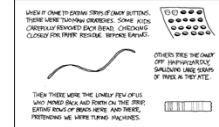
en.wikipedia.org/wiki/Turing_completeness

Turing Completeness

- A **Turing Machine** has an infinite tape of 1s and 0s and instructions that say whether to move the tape left, right, read, or write it
 - Can simulate any computer algorithm!
- A **Universal Turing Machine** is one that can simulate a Turing machine on any input
- A language is considered **Turing Complete** if it can simulate a **Universal Turing Machine**
 - A way to decide that one programming language or paradigm is just as powerful as another



Turing Machine by Tom Dunne



Xkcd comic "Candy Button Paper"

Ways to Remember the Paradigms

- | | |
|---|---|
| <ul style="list-style-type: none"> ▪ Functional <ul style="list-style-type: none"> ▫ Evaluate an expression and use the resulting value for something | <ul style="list-style-type: none"> ▪ Object-oriented <ul style="list-style-type: none"> ▫ Send messages between objects to simulate the temporal evolution of a set of real world phenomena |
| <ul style="list-style-type: none"> ▪ Imperative <ul style="list-style-type: none"> ▫ First <i>do this</i> and next <i>do that</i> | <ul style="list-style-type: none"> ▪ Declarative <ul style="list-style-type: none"> ▫ Answer a question via search for a solution |

www.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html

Summary

- Each paradigm has its **unique benefits**
 - If a language is Turing complete, it is equally powerful
 - Paradigms vary in efficiency, scalability, overhead, fun, "how" vs "what" to specify, etc.
- **Modern languages usually take the best from all**
 - E.g., Scratch
 - Can be functional
 - Can be imperative
 - Can be object-oriented!

