Andrew Plum

CS 210

1/17/2023

Week 2 Notes

1. Put CS 210 in subject line of emails
2. 60819
3. How many programming languages are there
   1. According to Wikipedia there are 700 esoteric programming languages
   2. Other sources say 245
4. Imperative program (C and C++)
   1. Good for
      1. Systems programming
      2. Portability
   2. Bad for
      1. Learning (very steep learning curve)
      2. Rapid application development
      3. Maintenance
   3. Trends
      1. Increased standardization
      2. Generative programming
5. Declarative languages
6. Object oriented languages
   1. Good for
      1. Domain modeling
      2. Developing reusable frameworks
   2. Bad for
      1. Learning (steep learning curve)
      2. Understanding (hard to keep systems well-structured)
      3. Semantics (no agreement)
   3. Trends
      1. Aspect-oriented programming
      2. Prototype-based programming (renaissance)
   4. Polymorphism
      1. A programming language feature that allows values of different data types to be handled using a uniform feature
      2. Good for
      3. Bad for
   5. Inheritance
7. Encapsulation and abstraction
8. Functional languages
   1. Good for
      1. Equational reasoning
      2. Declarative programming
   2. Bad for
      1. OOP
      2. Explicit concurrency
      3. Run time efficiency (although constantly improving)
   3. Trends
      1. Standardization: Haskell
      2. Extensions (concurrency, objects)
9. Lambda calculus
   1. The most prominent
   2. Good for
      1. Simple operational foundation for sequential programming languages
   3. Bad for
      1. Programming
   4. Trends
10. Logic programming
    1. Good for
       1. Searching (experts systems, graph &
    2. Bad for
11. Type systems
    1. Static typing
    2. Dynamic typing
12. Scripting languages
    1. Good for
       1. Rapid prototyping
    2. Bad for
13. Visual programming
    1. Good for
    2. Bad for
14. Programming languages
    1. Often grouped in four families
       1. Imperative (c is imperative)
       2. Functional
       3. Logic
       4. Object-oriented
15. Imperative languages
    1. Assignment
    2. Iteration - repeated execution of a set of statements
    3. Order of execution is critical
16. Functional languages
    1. Meta language (ML) and standard meta language (SML) and Lisp are examples of functional programming languages
    2. Single valued variables
    3. Heavy use of recursion
    4. No assignment, no iteration, just recursion
17. Logic languages
    1. Prolog is an example
    2. Programs expressed in rules of formal logic
18. Object-oriented languages
    1. Java definition for a kind of object that can store an integer and compute its factorial
    2. Java and c++ are examples of object oriented programming languages
    3. Allows many programmers to work independently of one another but still have their code work all together
    4. They are languages that are imperative plus
    5. Constructs to help programmers use “objects” - little bundles of data that know how to do things to themselves
19. Strengths and Weakness
    1. The different language groups show to advantage on different kinds of problems
    2. Decide for yourself at the end of the semester, after experimenting with them
    3. About those families
       1. There are many other language family terms (not exhaustive)
20. Stack oriented language
    1. Fourth is an example
    2. Utilizes the stack
21. APL
    1. Uses symbols not on the standard keyboard
22. The odd controversies
    1. Programming languages are the subject of many heated debates
       1. Partisan arguments
       2. Language standards
          1. Javas standards change on a consistent basis where as c has had the same standards for like 40 years
       3. Fundamental definitions
23. Language partisans
    1. There is a lot argument about the relative merits of  different languages
24. Language standards
    1. The documents that define language standards are often drafted by interns committees
    2. Can be a slow, complicated and rancorous process
25. Basic definitions
    1. Some terms refer to fuzzy concepts: all those language family names, for example
    2. No problem, just remember they are fuzzy
       1. Bad
       2. Good
26. Intriguing evolution
    1. Programming languages are evolving rapidly
       1. New languages are being invented
27. New languages
    1. A clean state: no need to maintain compatibility with an existing node of code
    2. But never entirely new anymore: always using ideas from earlier designs
    3. Some become widely used others do not
    4. Whether widely used or not
28. Widely used: Java
    1. Quick rise to popularity since 1995 release
    2. Java uses many ideas from c++ plus some from Mesa, Modula, and other languages
    3. C++ uses most of C and extends it with ideas Simula 67, Ada, Clu, ML and Algol 68
    4. C was derived from B, which
29. Not widely used: Algol
    1. One of the earliest languages: Algol
30. Dialects
    1. Experience with languages reveals their design weakness
31. Some dialects of Fortran
    1. Fortran 1-4
    2. Fortran 66
    3. Fortran 77
    4. Deviations in each implementation
    5. Parallel processing
       1. HPF
       2. Fortran M
       3. Vienna Fortran
32. The connection to programming practice
    1. Languages influence programming practice
       1. A language favors a particular programming style
       2. T
33. Languages influences programming practice
    1. Languages often strongly favor a particular style of programming
       1. Object oriented language: a style making heavy use of objects
       2. T
       3. T
       4. T
34. Fighting the language
    1. Languages favor a particular style
35. Non-object oriented Java
    1. Java more than c++ tries to encourage you to adopt an object
36. Functional pascal
37. Functional
38. Programming experience influences language design
    1. Corrections to design
39. Language evolution drives and is driven by hardware evolution
    1. Call stack support languages with recursion
    2. Parallel architectures - parallel architectures - parallel languages led to Java
40. Theory of formal languages is a core mathematical area of computer science
    1. Regular grammars, finite-state automata - lexical structure of programming languages, scanner in a compiler
    2. Context free grammar, push down automata
    3. Turing machines
41. Turing equivalence
    1. Languages have different strengths, but fundamentally they all have the same power
    2. And all have the same power as various mathematical models of computation
       1. Problems solvable by Turing machine
          1. Problems solvable by java = problems solvable by Fortran
       2. Problems solvable by lambda calculus
          1. Problems solvable by Turing machine = problems solvable by lambda calculus
    3. Church-Turing thesis: this is what “computability” means