1 - Intro to Programming Languages

Supplemental

Lecture

- Languages we will earn: Python (fluency), Haskell, and Prolog (basics)

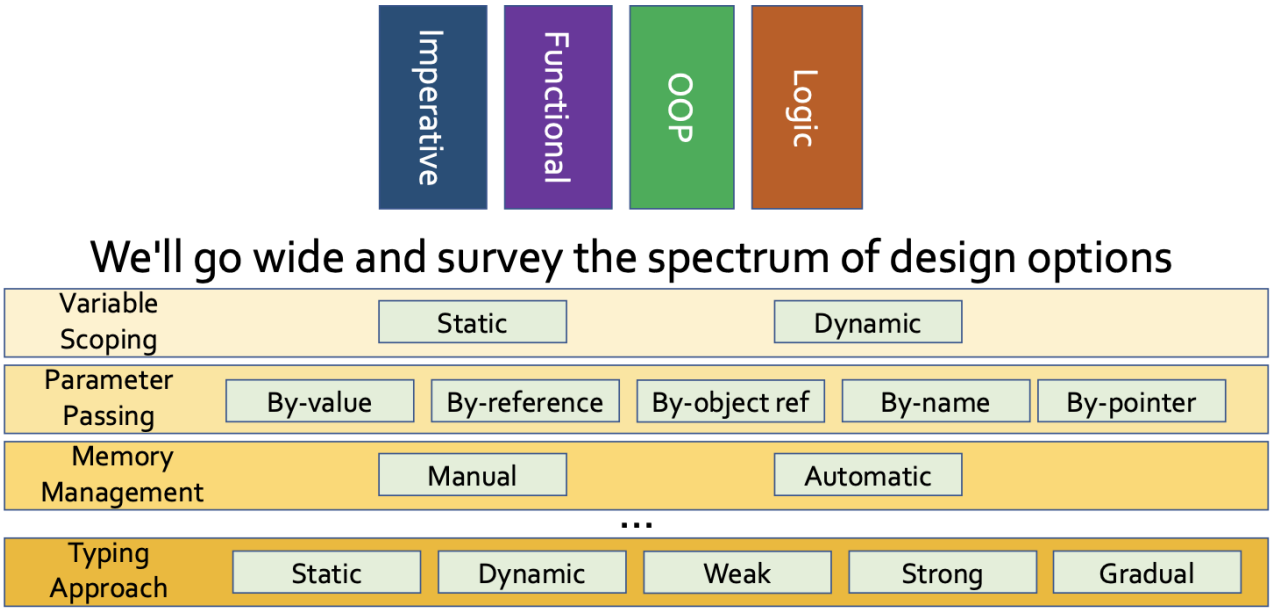
Programming Paradigms

- Imperative (C)
  - Statements, loops, mutable variables
- OOP (Smalltalk)
  - Object-based
  - Objets send messages and have properties
- Functional (Haskell)
  - Math-like functions
  - There are no statements or iterations, only equations
- Logical (Prolog)
  - Define a set of facts and rules
  - Query rules

Language choices can include:

- Type checking
  - Static, dynamic
- Parameter passing
  - By-value, reference, pointer, object reference
- Scoping
  - Lexical or dynamic
- Memory management

- Manual or automatic



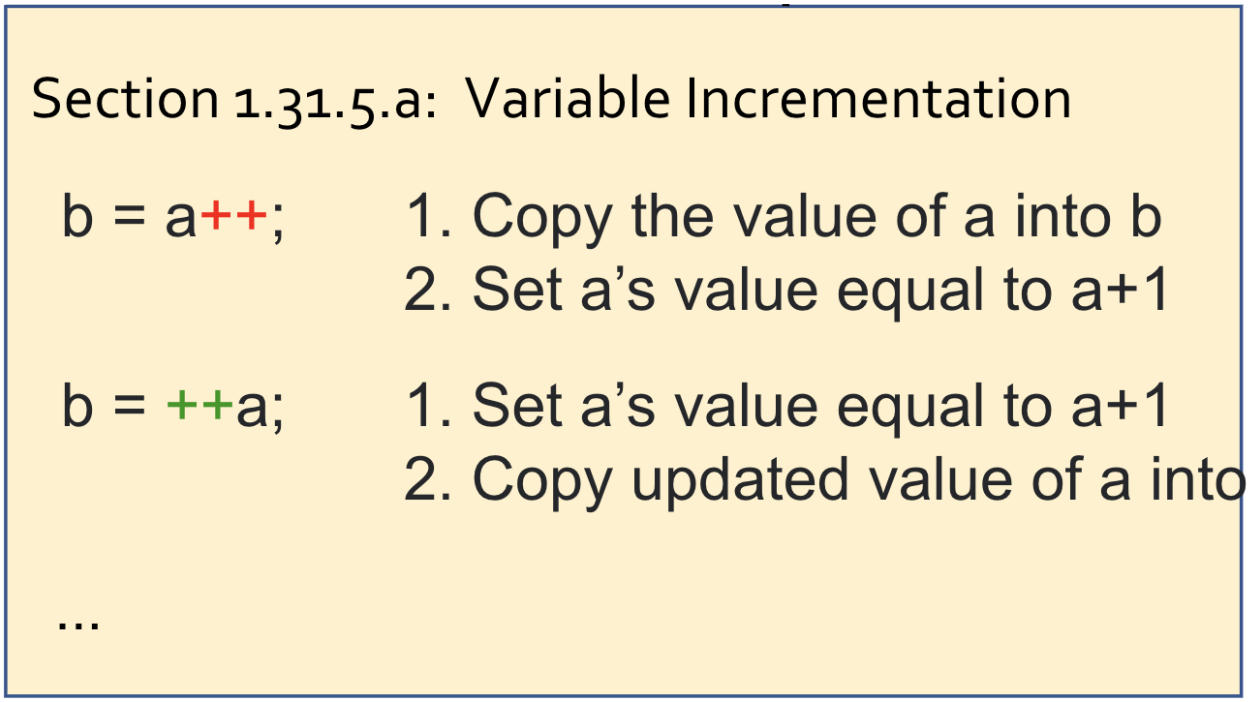
Parameter Passing

- By-value
  - The function creates a copy of the object
  - No changes saved to original variable
- By-reference
  - The function takes the variables address
  - Changing the reference's property changes the original variable's properties
  - BUT, assignment also changes the original bc its an alias
  - I.e. if we take input s, then set s = new …
  - The original changes to the new object
- By-pointer
  - The function takes a pointer to the address
  - So any pointer based property changes willl reflect
  - But new assignment only redirects the pointer so the og var does not change
  - E.g., func takes input s, s→prop = something changes the og var

    - But s = new prop does NOT change the OG
- By object reference
    - Same as by-pointer BUT no object dereferencing

Compiling and Linking

Syntax spec - specifies the syntax of a language (Grammar)

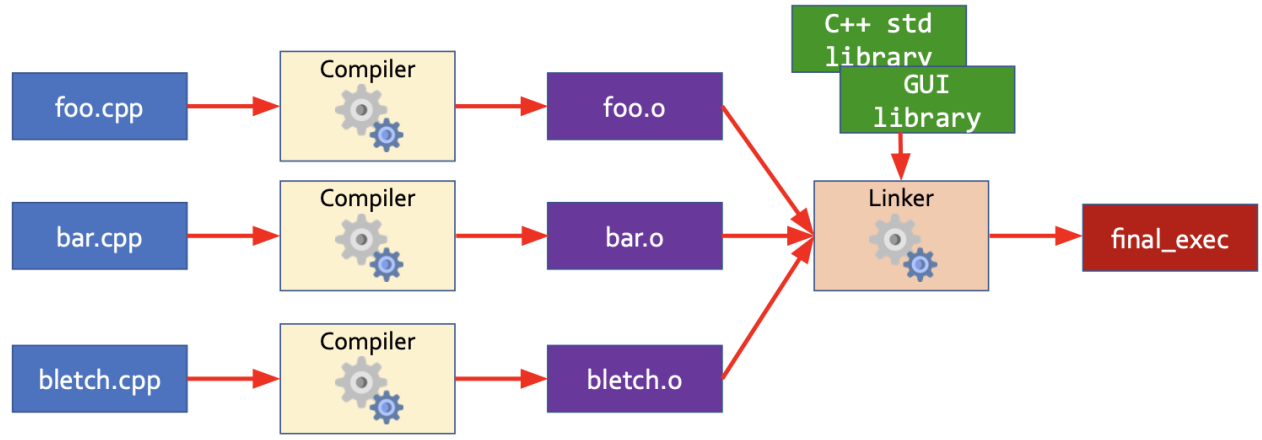


Compiler - translates program source code into object modules (either machine language or bytecode for an interpreter)

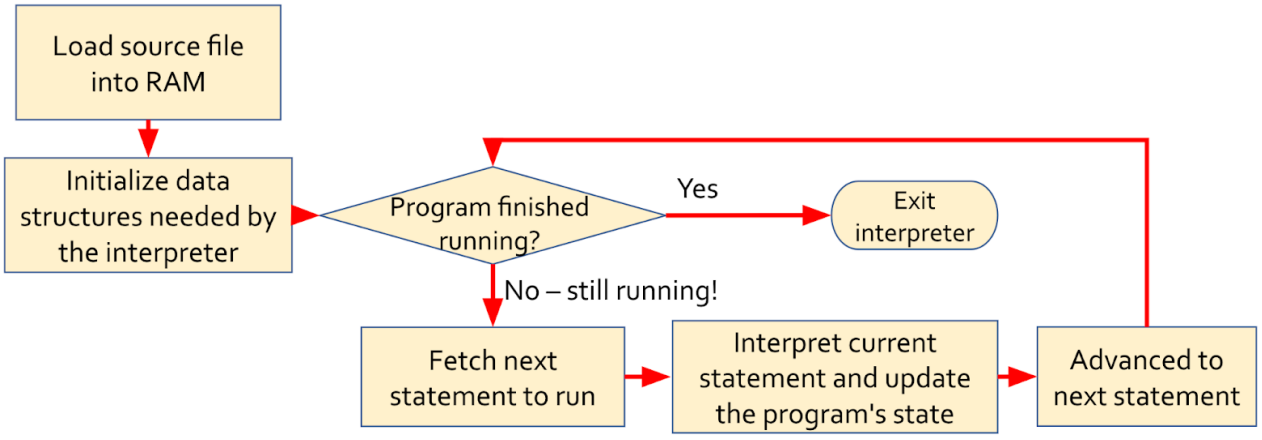
- **Lexical analyzer - breaks source code into lexical tokens/units (keywords like for, while, etc.)**
    - E.g. if you type "fro" instead of "for" it will throw
- **Parser checks lexical tokens and validates for syntax; usually listed in EBNF (extended Backus–Naur form); and creates an abstract syntax tree (AST)**
    - E.g. missing closing parentheses throws an error
- **Semantic analyzer takes AST and checks semantic validity and updates the AST nodes and outputs and**

annotated pare tree
    - E.g., checks multiplying a string with an int
- **Intermediate representation generator creates an abstract/agnostic representation of the tree from the APT and outputs an intermediate representation**
    - May look like assembly
- **Code generator - converts to machine code or bytecode into an object file for an interpreter or JIT compiler**
    - Architecture specific e.g., x86, MIPS, etc.

Linker - combines multiple object modules and libraries into single executable file or lib



Interpreter - directly executes program statements without compiling them to machine language first



"/1-intro_lecture.pptx.pdf" is not created yet. Click to create.

📌

SUMMARY