Overview of Compilers
Part 2

Some Data Structures

- Symbol table
- Literal table
- Parse tree

Symbol Table

- Identifiers are **names** of variables, constants, functions, data types, etc.
- Store information associated with identifiers
  - Information associated with different types of identifiers can be different
    - Information associated with variables are name, type, address, size (for array), etc.
    - Information associated with functions are name, type of return value, parameters, address, etc.
Symbol Table (cont’d)

- Accessed in every phase of compilers
  - The scanner, parser, and semantic analyzer put names of identifiers in symbol table.
  - The semantic analyzer stores more information (e.g. data types) in the table.
  - The intermediate code generator, code optimizer and code generator use information in symbol table to generate appropriate code.
- Mostly use hash table for efficiency.

Literal table

- Store constants and strings used in program
  - reduce the memory size by reusing constants and strings
- Can be combined with symbol table

Parse tree

- Dynamically-allocated, pointer-based structure
- Information for different data types related to parse trees need to be stored somewhere.
  - Nodes are variant records, storing information for different types of data
  - Nodes store pointers to information stored in other data structure, e.g. symbol table
Error Handling

• Error can be found in every phase of compilation.
  – Errors found during compilation are called static (or compile-time) errors.
  – Errors found during execution are called dynamic (or run-time) errors.
• Compilers need to detect, report, and recover from error found in source programs.
• Error handlers are different in different phases of compiler.

Cousins of Compilers

• Linkers
• Loaders
• Interpreters
• Assemblers
• Language Translators (e.g., Google Translate)
• AI Intelligent Agents (“chatbots”)
• Layout Engines (e.g., html)
• Spam Filters
• Search & Replace functionality
• Writing Evaluators