Lecture 18:

- Quiz 4
- Semantic Analysis (cont)

Announcements:

- HW-3 due Mon
The goal of static analysis is to:

- Detect errors due to type issues, e.g.:

  ```
  x = 0 + "1"; // int + string not allowed
  if (42 <= true) {
    x = 1;
  }
  ```

- Detect “use before def” errors, e.g.:

  ```
  int x = 42 + y; // y not defined
  if (x > 42) {
    int y = x + 1;
  }
  else {
    x = y; // y not defined in this block
  }
  ```

- Detect function call errors, e.g.:

  ```
  int add(int x, int y) {
    return x + y;
  }
  void main() {
    int r1 = add(1, 2, 3); // wrong number of args
    int r2 = add(3.14, 1); // wrong argument types
    bool r3 = add(1, 2); // wrong return type
  }
  ```

- plus more ...
Examples of other kinds of errors found during static analysis

- duplicate function names
- duplicate struct names
- parameters with duplicate names and fields with duplicate names
- variable shadowing
- main function not defined
- and so on

Type errors often based on a set of typing rules (aka "judgements")

- the rules define how types can be "inferred" (inference rules)
- statements or expressions that violate the rules have type errors
Basic Idea of Semantic Analysis (HW-4)

• navigate the AST using the Visitor pattern
  • during navigation infer types and look for errors

Given this code ...

1: int x = 10;
2: int r = 0;
3: while (x > 0) {
4:    r = r + x;
5:    x = x - 1;
6: }

1. For “int x = 10;”
   • check and infer rhs type, compare against declared type, store x’s type
2. Similarly for “int r = 0;”
3. For “while (x > 0) ... ”
   • ensure x is defined and compatible with 0 (both ints)
   • check each body statement ...
4. For “r = r + x;”
   • ensure in rhs that r and x are defined and types are compatible for +
   • ensure lhs is defined and result type (int) is compatible with lhs (r) type
5. etc.

Example suggests we need to keep track of names and their types!

• we’ll do this using a “symbol table”
• data structure for managing bindings (id -> type) in environments