Lecture Notes 9

Expressions and Statements

- Expression – Purely mathematical value that have no side effects
  - Example of an expression representing 3 added to x, x is not changed in the evaluation
    `x + 3`
- Statement – A command executed for the side effect, in theory have no value
  - Example of a statement assigning x to 3, the value of x changes to 3
    `x = 3`
- C Statements/Expressions – Many statements in C (those executed for side effects) actually have a value as well. C has mix of theoretical Statement and Expression
  - Example of a C expression with side effect, the value of statement is x prior to incrementing, side effect increments x by 1.
    `x++`

Arithmetic Expressions

- Unary Operators – Operator only has one operand
  - Example of logical inversion in C
    `!x`
- Binary Operators – Operator has two operands
  - Example of modulus in C
    `x % 6`
- Ternary Operators – Operator has three operands
  - Example of C ternary operator to find min of x and y
    `x < y ? x : y`
- Infix Operators – Operator that appears in between the operands
  - Example of infix addition
    `x + y`
- Prefix Operators – Operator appears before the operands
  - Example of Lisp addition
    `(+ x y)`
- Postfix Operators – Operator appears after the operands
  - Example of C post increment
    `x++`
- Operator Precedence Rules – Defines the order in which operators of different precedence levels are evaluated
- Identity Operator – The unary + preceding the operand

- C-based Language Precedence

Highest	postfix `++`, `--`
	prefix `++`, `--`, unary `+`, `-`
	`*`, `/`, `%`
Lowest	binary `+`, `-`

- Associativity – Rules for order of evaluation within same precedence level
- Left-to-right Associativity – Evaluated from the left to right is most common in languages (exponentiation may be reverse)
- C-based Language Associativity

Left:	`*`, `/`, `%`, binary `+`, `-`
Right:	`++`, `--`, unary `+`, `-`

- Parentheses `()` – Used to change precedence and associativity rules
- Referential Transparency – Property of a program if any two expressions with the same value can be substituted for one another

Operator Overloading

- Operator Overloading – Reuse of the same operator to refer to different program constructs.
  - Operators `+`, `-`, `*`, `/` – Operators are almost always overloaded to work for integer and float, but may be allowed additional meanings (concatenation of strings)
  - Operator Overloading and Classes – Some languages support operator overloading for classes