Expressions & Assignment

Expressions

- Operators
  - Unary, binary, ternary
  - Infix vs prefix
- Design
  - Operator precedence rules
  - Associativity rules
  - Operand evaluation
  - Side effects
  - Overloading
  - Mixed-mode expressions

Operator Order

- Precedence
  - Rules for arithmetic operators
  - Unary minus
  - Number of levels
  - Rules
- Associativity
  - Same precedence
  - Typically L->R
  - Exponentiation typically R->L
  - Rules
  - Optimization
- Grouping (parentheses)
Variations

- APL
  - No precedence
- Ruby
  - Operators syntactic sugar for method calls
- Lisp
  - Functional hence prefix
  - Conditional expressions
    - E.g. C
- Operator overloading
  - Predefined
  - User defined
  - Precedence?
Operand Evaluation

- Value(s) fetched prior to execution of operator
  - When?
  - Subexpressions
  - Functions
    - Also evaluation of parameters
- Side effects
  - Evaluation of operand changes machine state
  - Unpredictability
    - Solutions
      - Disallow side effects
        - Hard
      - Specify evaluation order
        - Java
        - Optimization

```c
int a = 5;
int fun1()
    
    a = 17;

return a;

} /* end of fun */

test main()
    
    a = a + fun1();

} /* end of main */
```
Referential transparency

- Substitutability of equivalent expressions
- Result of function (operation) depends solely on value of operands (not even evaluation order)
- Pure functional languages

Advantages
- Reasoning
- Optimization

Type Conversion

- Narrowing vs widening
- Mixed-mode vs single-mode expressions
- Coercion
  - Implicit conversion
  - vs no implicit conversion
  - Eg.
    - Ada - no conversions
    - ML, F# - no coercion
    - Java - limited coercions
    - C - most coercions
- Explicit coercion
  - Cast
- Errors
  - Overflow/underflow
  - Division by zero
Relational & Boolean Expressions

- Relational
  - Result typically boolean
  - Operands ordinal types
  - Equality operators
    - \( = \)
    - \( == \) vs \( =\= \)
    - \( \neq \) vs \( /= \)
    - \( <> \)

- Boolean
  - Operands boolean, result boolean
  - and, or, not
  - Exclusive or?
  - Bit operations (\( \land \) and \( \lor \) in C vs Java)
  - No boolean type?
  - Operator precedence
  - Short-circuit
  - Operand evaluation
  - Guards

Assignment

- Token
- Compound assignment
  - E.g. C
- Unary assignment
  - E.g. C: prefix & postfix
  - Side effects
- Conditional assignment
  - E.g. Perl
- Assignment as operator
  - Value is value assigned to lhs
  - Side effects – leads to lack of readability
  - \( = vs \circlearrowleft in C \) conditionals
- Multiple assignment
  - And side effects
- Mixed-mode assignment (coercion)
- Functional languages
  - \( \text{val} \) and \( \text{let} \)
```plaintext
sum += value;
sum = sum + value;

sum = ++count;
sum = count++;

count = count + 1;
sum = count;

count ++;

if (flag ? $count1 : $count2) = 0;
```

```plaintext
while ((ch = getchar()) != EOF) { ... }

a = b + (c = d / b) - 1

Assign d / b to c
Assign b + d to temp
Assign temp - 4 to a

sum = count = 0;

if (x < y) ...
if (x > y) ...
```

```plaintext
(First, Second, Third) = (23, 40, 60);
(First, Second) = (Second, First);

val cost = quantity * price;
```