Lecture 2

- Lecture 2

  - Agenda
    - Questions regarding Chapter 1 or 2?
    - Relevant/complicated subject from Chapter 3
    - Exercise: Compute the value of Pi using an infinite series

  - Chapter 3
    - The Main entry point
      - Create console app and explain its part
      - Exit code signals result to operating system/other apps
      - PowerShell.exe or cmd.exe executes/composes programs
    - Main method is a method like any other
    - Explain Main method signatures as a reasoning tool using sets Data types in Math vs. C#
    - Ties into Common Type System data types and the Venn diagram
    - Signature is a way of communicating intent
      - `double balance = account.ComputeInterest(int days, double rate);`
      - Imagine input, output types being classes, struct, enums also

    - Widening and narrowing data type conversions
      - Widening
        - Implicit widening happens when there's no risk of loss of data
        - Smaller value is stored within a larger data type variable
      - Narrowing
        - Larger value is stored within a smaller data type variable

        -
          ```
          int Add(int x, int y) { return x + y; }
          // short is a 16 bit signed value -- causes overflow exception
          short s = Add(30000, 30000);
          ```
        - Explicit narrowing
          ```
          // no compiler error but result is wrong
          short s = (short) Add(30000, 30000);
          ```
        - Illustrate overflow with a circular number line (or a dial clock)

    - Checked/unchecked overflows
      - Rarely used in practice
      - Detects overflow and throws exception at runtime

    - Implicitly typed local variables (see ILSpy)
      - Pros and cons

    - Boolean truth tables (illustrates boolean expression short-curcuiting)
      - Because only two values, you can construct table of all combinations
      - Illustrate for * by a good old multiplication table
      -

Or	F	T
F	F	T
T	T	T

And	F	T
F	F	F
T	F	T

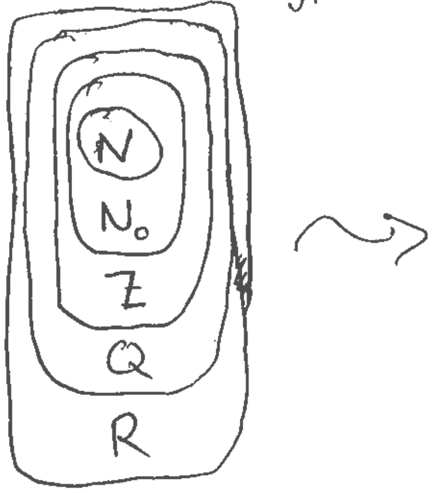
Not	F
T	F
F	T

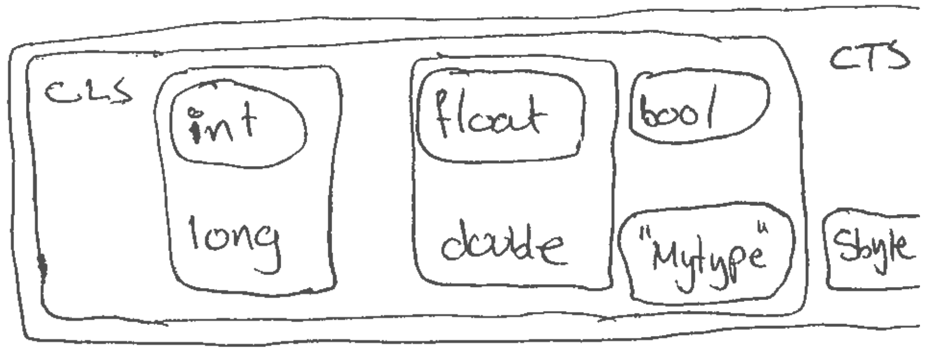
      - Example: `if (age == 30 || name == "Fred") { ... }`
        - Evaluate individual parts
          - `bool t1 = age == 30;      // to True, for instance`
          - `bool t2 = name == "Fred"  // to False, for instance`
          - so expression becomes
            `bool r = True || False;   // True`
            lookup result in truth table
          - Imagine t1 and t2 being method calls. Then t2 wouldn't be called
          - A way to make programs run faster, but it can trick you

Data types in Math vs. C#

Natural numbers	$N = \{1, 2, 3, \dots\}$	N_0 if zero is included
Integer numbers	$Z = \{-2, -1, 0, 1, \dots\}$	
Rational numbers	$Q = \{-\frac{1}{1}, \frac{22}{7}, \dots\}$	
Real numbers	$R = \{-2, 0, 3.14, \frac{22}{7}\}$	uncountable

Each number type is a subset of another





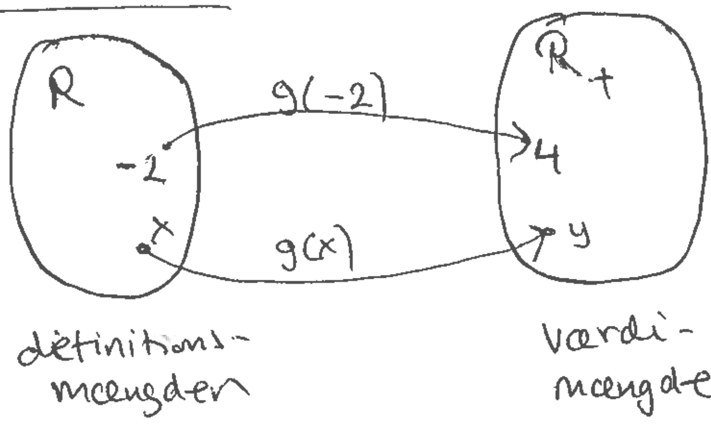
Page 86

$f(x) = x$ $dm(f) = R$, $vm(f) = R$ identity function

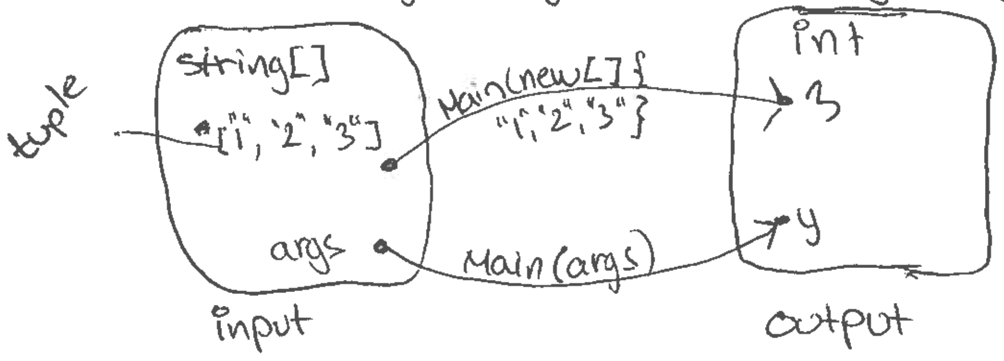
$g(x) = x^2$ $dm(g) = R$, $vm(g) = R_+$ $- \cdot - = +$

$f : R \to R$ $\leadsto$ f : input $\to$ output type signatures

$g : R \to R_+$ Main : String[] $\to$ int



int Main(string[] args) {return args.Length;}



Programming langs. are math notation inspired, but usually more wordy / less terse.