§7.6-7.13: Applications

•*devo*

20 Oct 2005
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders:

- ***journals in folder***

TRINITY WESTERN UNIVERSITY

What's on for today (7.6-7.13)

- Application: pseudo-random number generator
  - Persistent variable (seed) internal to library
  - Initialization in body of implementation file
- Application: substitution cipher
  - Designing public interface (DEF)
  - Using private helper functions
- Application: fractions (time permitting)
  - Designing an ADT as a library

Application: Random numbers

- A random number (from a uniform distribution) is chosen such that every number within the range is equally likely to be chosen:
  - Uniform distribution on [0..1]
- Making things truly random (high entropy) is very difficult!
  - Hardware random-number generators:
    - Measure radioactive decay of isotopes
    - Brownian motion of particles in a suspension (air)
  - Software pseudo-random number generators

Pseudo-random number generator

- A pseudo-random number generator applies some math operations to the last number generated to get the next number
  - Start with a seed number
  - Hopefully it's “random enough”
  - But really it's completely deterministic:
    - If we start again with the same seed, we'll always get the same sequence of “random” numbers
- e.g., seed=0.10: generates
  - 0.72, 0.23, 0.19, 0.93, 0.54, 0.77, 0.11, ...

DEF: pseudo-random num library

- We only need Random() as a public procedure:

```
DEFINITION MODULE PseudoRandom;

PROCEDURE Random () : LONGREAL;
    (* returns a random number between 0 and 1 *)

PROCEDURE InitSeed (x : LONGREAL);
    (* initialize the number generator seed *)

END PseudoRandom.
```

- InitSeed provides a way for the user to manually set the seed.

IMP: pseudo-random num library

```
IMPLEMENTATION MODULE PseudoRandom;

FROM LongMath IMPORT
    exp, ln, pi;
VAR
    seed : LONGREAL;   (* persistent across calls to Random() *)

PROCEDURE InitSeed (x : LONGREAL);
    (* accessor (set) function for seed *)
BEGIN
    seed := x;
END InitSeed;
```

IMP: PseudoRandom, cont.

```
PROCEDURE Random (): LONGREAL;
BEGIN
    (* try to scramble up seed as much as possible *)
    seed := seed + pi;
    seed := exp (7.0 * ln (seed));

    (* just keep fractional part, in range 0..1 *)
    seed := seed – LFLOAT (TRUNC (seed));
    RETURN seed;
END Random;

BEGIN
    seed := 0.0;        (* initialize in body of module *)
END PseudoRandom.
```

Online test of PseudoRandom

- (demo in Stonybrook of PseudoRandomTest)

- Evaluating "randomness":
  - Graphical evaluations: plot points (x,y) where both coordinates are from Random()
  - Check for dense spots, sparse spots in 1x1 square
  - M2 has a graphics library, but it's beyond the scope of this class

Cryptography example

- Cæsar substitution cipher:
  - Key: e.g., QAZXSWEDCVFRTGBNHYUJMKIOLP
  - Cleartext: input text to encrypt
  - Ciphertext: output encrypted text
  - Encoding: replace each letter in source with corresponding letter from code key
  - Decoding: same, using the decode key
- ROT13 was an example of a substitution cipher
  - Key: NOPQRSTUVWXYZABCDEFGHIJKLM

Write a Substitution cipher library

- What public interface do we want for the library?

```
DEFINITION MODULE Substitution;
TYPE CodeString = ARRAY [0..25] OF CHAR;

PROCEDURE Encode (src: ARRAY OF CHAR;
  VAR dst: ARRAY OF CHAR; key: CodeString);

PROCEDURE Decode (src: ARRAY OF CHAR;
  VAR dst: ARRAY OF CHAR; key: CodeString);

END Substitution.
```

Implementing Substitution

- In the implementation it is handy to have some helper functions for internal use: these will not be exported:

```
IsLetter (ch: CHAR) : BOOLEAN;
    (* check if it's a letter or some other character *)
AlphaPos (ch: CHAR) : CARDINAL;
    (* index of a letter in the range 0..25 *)
DecodeKey (enckey: CodeString; deckey: CodeString);
    (* create a decode key from an encoding key *)
```

- How to implement these?

TODO items

- Homework due tomorrow: 6.11 #28
- Quiz ch7 tomorrow
- Lab #6 next week: 7.14 #(22 / 32 / 37)
- Reading: through end of book for tomorrow
- 140 Final next week W-Th (two parts)