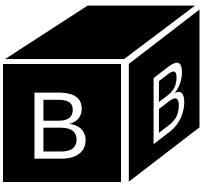


Chapter *14*

Random Numbers

```
DEFINITION PboxRandom;
    CONST
        seedLimit = 2147483647;

    PROCEDURE Int (n: INTEGER): INTEGER;
    PROCEDURE Randomize;
    PROCEDURE Real (): REAL;
    PROCEDURE SetSeed (n: INTEGER);

END PboxRandom.
```

Figure 14.1
The interface of module PboxRandom.

PROCEDURE **SetSeed** (n: INTEGER);
Pre
0 < n 20
n < seedLimit 21
Post
The random number seed is initialized to n.

PROCEDURE **Randomize**
Post
The random number seed is initialized to a value derived from the system clock.
Two calls to Randomize should be separated by more than one second to guarantee different values of seed.

PROCEDURE **Real** (): REAL;
Post
Returns a random real between 0.0 and 1.0.

```
MODULE Pbox14A;
    IMPORT Dialog, PboxRandom, PboxStrings, StdLog;
    VAR
        d*: RECORD
            seed*: INTEGER;
        END;

    PROCEDURE SetSeed*;
    BEGIN
        IF (0 < d.seed) & (d.seed < PboxRandom.seedLimit) THEN
            PboxRandom.SetSeed(d.seed)
        ELSE
            StdLog.String("Seed must be greater than 0 and less than 2147483647."); StdLog.Ln
        END
    END SetSeed;

    PROCEDURE Randomize*;
    BEGIN
        PboxRandom.Randomize
    END Randomize;
```

Figure 14.2
A procedure that prints ten random real numbers to the Log.

```
    PROCEDURE Display*;
    VAR
        i: INTEGER;
        x: REAL;
        realString: ARRAY 8 OF CHAR;
    BEGIN
        FOR i := 1 TO 10 DO
            StdLog.String("i = "); StdLog.Int(i);
            x := PboxRandom.Real();
            PboxStrings.RealToString(x, 5, 3, realString);
            StdLog.String("    x = "); StdLog.String(realString); StdLog.Ln
        END;
        StdLog.Ln
    END Display;

BEGIN
    d.seed := 1
END Pbox14A.
```

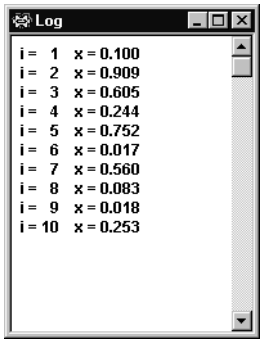
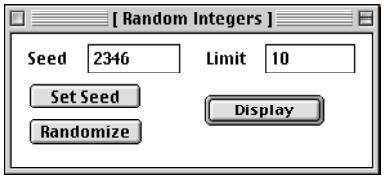


Figure 14.3
The output of procedure Display of Figure 14.2.



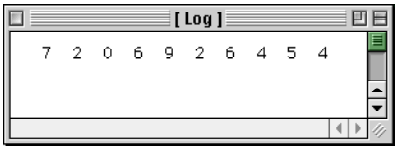


Figure 14.4
The output for the procedure of Figure 14.5.

```
MODULE Pbox14B;
    IMPORT Dialog, PboxRandom, PboxStrings, StdLog;
    VAR
        d*: RECORD
            seed*: INTEGER;
            limit*: INTEGER;
        END;

    PROCEDURE SetSeed*;
    BEGIN
        IF (0 < d.seed) & (d.seed < PboxRandom.seedLimit) THEN
            PboxRandom.SetSeed(d.seed)
        ELSE
            StdLog.String("Seed must be greater than 0 and less than 2147483647."); StdLog.Ln
        END
    END SetSeed;

    PROCEDURE Randomize*;
    BEGIN
        PboxRandom.Randomize
    END Randomize;
```

Figure 14.5
A procedure that prints ten random integers to the Log.

```
    PROCEDURE Display*;
    VAR
        i: INTEGER;
        m: INTEGER;
    BEGIN
        IF (0 < d.limit) & (d.limit < PboxRandom.seedLimit) THEN
            FOR i := 1 TO 10 DO
                m := PboxRandom.Int(d.limit);
                StdLog.Int(m)
            END;
            StdLog.Ln
        ELSE
            StdLog.String("Limit must be greater than 0 and less than 2147483647."); StdLog.Ln
        END
    END Display;

BEGIN
    d.seed := 1;
    d.limit := 0
END Pbox14B.
```

PROCEDURE **Int** (n: INTEGER): INTEGER;
Pre
0 < n 20
n < seedLimit 21
Post
Returns a random integer in the range 0..n-1.

```
REPEAT
    Statement1
UNTIL Condition1
```

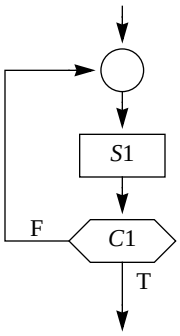
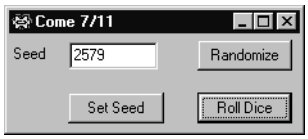


Figure 14.6
The flowchart for the REPEAT statement.



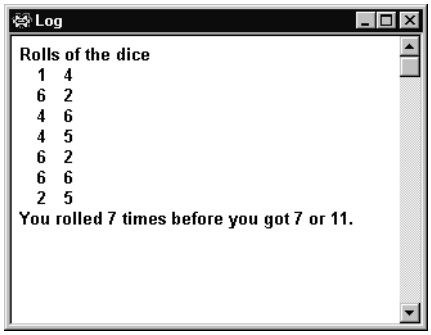


Figure 14.7
The output for the procedure in Figure 14.8.

```
MODULE Pbox14C;
    IMPORT Dialog, PboxRandom, PboxStrings, StdLog;
    VAR
        d*: RECORD
            seed*: INTEGER;
        END;

    PROCEDURE SetSeed*;
    BEGIN
        IF (0 < d.seed) & (d.seed < PboxRandom.seedLimit) THEN
            PboxRandom.SetSeed(d.seed)
        ELSE
            StdLog.String("Seed must be greater than 0 and less than 2147483647."); StdLog.Ln
        END
    END SetSeed;

    PROCEDURE Randomize*;
    BEGIN
        PboxRandom.Randomize
    END Randomize;
```

Figure 14.8
A procedure that simulates rolls of a pair of dice.

```
    PROCEDURE RollDice*;
    VAR
        die1, die2: INTEGER;
        sum, numRolls: INTEGER;
    BEGIN
        numRolls := 0;
        StdLog.String("Rolls of the dice"); StdLog.Ln;
        REPEAT
            die1 := PboxRandom.Int(6) + 1;
            die2 := PboxRandom.Int(6) + 1;
            INC(numRolls);
            sum := die1 + die2;
            StdLog.Int(die1); StdLog.Int(die2); StdLog.Ln
        UNTIL (sum = 7) OR (sum = 11);
        StdLog.String("You rolled "); StdLog.Int(numRolls);
        StdLog.String(" times before you got 7 or 11."); StdLog.Ln
    END RollDice;

BEGIN
    d.seed := 1
END Pbox14C.
```

$z_{n+1} = az_n \text{ MOD } m$

Example If you select values of (17, 5) for (*m*, *a*), and the current value of seed z_n is 11, then the next seed is computed as

$$\begin{aligned} z_{n+1} &= az_n \bmod m \\ &= 5 \cdot 11 \bmod 17 \\ &= 55 \bmod 17 \\ &= 4 \end{aligned}$$

The 20 successive seed values from the (17, 5) Lehmer generator are

11 4 3 15 7 1 5 8 6 13 14 2 10 16 12 9 11 4 3 15

with an initial seed of 11. ▮

Example The (17, 13) Lehmer generator produces the sequence
4 1 13 16 4 1 13 16 4 1 13

starting from 4.

```
MODULE PboxRandom;
    IMPORT Dates;
    CONST
        multiplier = 48271;
        modulus = 2147483647;
        quotient = modulus DIV multiplier;
        remainder = modulus MOD multiplier;
        seedLimit* = modulus;
    VAR
        seed: INTEGER;

    PROCEDURE ComputeNextSeed;
        VAR
            low, high: INTEGER;
    BEGIN
        low := seed MOD quotient;
        high := seed DIV quotient;
        seed := multiplier * low - remainder * high;
        IF seed <= 0 THEN
            seed := seed + modulus
        END
    END ComputeNextSeed;
```

Figure 14.9
An implementation of the standard (2147483647, 48271) Lehmer random number generator.

```
PROCEDURE Int* (n: INTEGER): INTEGER;
BEGIN
    ASSERT(0 < n, 20);
    ASSERT(n < seedLimit, 21);
    ComputeNextSeed;
    RETURN SHORT(ENTIER(seed / modulus * n))
END Int;

PROCEDURE Randomize*;
    VAR
        date: Dates.Date;
        time: Dates.Time;
        i: INTEGER;
BEGIN
    Dates.GetDate(date); Dates.GetTime(time);
    seed := 86400 * Dates.Day(date) + 3600 * time.hour + 60 * time.minute
        + time.second; (* Elapsed time this year in seconds *)
    FOR i := 0 TO 7 DO
        ComputeNextSeed
    END
END Randomize;
```

```
    PROCEDURE Real* (): REAL;
    BEGIN
        ComputeNextSeed;
        RETURN seed / modulus
    END Real;

    PROCEDURE SetSeed* (n: INTEGER);
        VAR
            i: INTEGER;
    BEGIN
        ASSERT(0 < n, 20);
        ASSERT(n < seedLimit, 21);
        seed := n MOD modulus;
        FOR i := 0 TO 7 DO
            ComputeNextSeed
        END
    END SetSeed;

BEGIN
    Randomize
END PboxRandom.
```

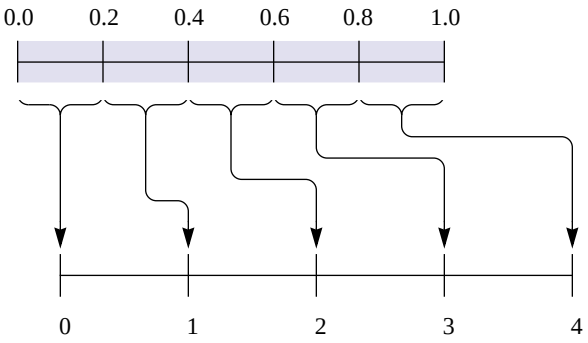


Figure 14.10
The transformation PboxRandom.Int makes from the real number line to the integer number line.