```
MODULE TestGCAlloc1;

  (*
    Benchmark on uniformly distributed, random large allocations.

    Origin:   https://github.com/D-Programming-
Language/druntime/blob/master/benchmark/gcbench/rand_large.d

    Copyright:   Copyright David Simcha 2011 - 2011.
    License:   Boost License 1.0   (See accompanying file LICENSE or copy at
http://www.boost.org/LICENSE_1_0.txt)
    Authors:   David Simcha
    Contributors:   Romiras 2013 (translated to Component Pascal + some modifications)
                    Robert Campbell, Aug-2013,
                      Sets random seed to a controlled valued
                      Outputs allocation performance statistics.
  *)

  IMPORT O := ObxRandom, Out;

  CONST
    nIter = 100;
    minSize = 1024 * 1000H;    (*  4 MByte  *)
    maxSize = 128 * minSize + 1;    (*  512 MByte  *)

  TYPE
    Item = POINTER TO ARRAY OF BYTE;
    Array = ARRAY 1024 OF Item;

  VAR
    array : Array;
    item: Item;

  PROCEDURE UnifomInt (a, b: INTEGER): INTEGER;
    VAR r : REAL;
  BEGIN
    ASSERT(a < b, 22);
    r := O.Uniform();
    RETURN SHORT(ENTIER(a + r * (b - a)))
  END UnifomInt;

  PROCEDURE Do*;
    VAR i, j, okCnt, badCnt, size, alloc, maxAlloc: INTEGER;
  BEGIN
    O.InitSeed(314159);
    okCnt  :=  0;
    badCnt  :=  0;
    maxAlloc  :=  0;
    FOR i := 1 TO nIter DO
      alloc  :=  0;
      FOR j := 0 TO LEN(Array) - 1 DO
        size := UnifomInt(minSize, maxSize);
        NEW(item, size);
        IF item = NIL THEN INC(badCnt) ELSE INC(okCnt); INC(alloc, size) END;
        array[j] := item
      END;
      maxAlloc := MAX(maxAlloc, alloc);
      Out.String("Iteration "); Out.Int(i, 0); Out.Ln
    END;
    Out.String("Good allocations "); Out.Int(okCnt, 0); Out.Ln;
    Out.String("Failed allocations "); Out.Int(badCnt, 0); Out.Ln;
    Out.String("Maximum allocation "); Out.Int(maxAlloc, 0); Out.Ln
  END Do;
```

```
END TestGCAlloc1.
```

- DevDebug.Unload
- TestGCAlloc1.Do