```
------------------------------------------------------------------------
   BenchMaker1 - BM1 V1.6.4 (Windows) - (C) 2001-2014 by Jozo Dujmovic
   Generator of sequences of random sosurce C++ benchmark programs
------------------------------------------------------------------------
   Generated random programs contain functions and the main program.
   The size of all programs is adjustable and programs can be executed.
   This version of BM1 can be used in both interactive and engine mode.
   Note: use wide screen to see all messages and programs
------------------------------------------------------------------------

BM1 operation modes:
   1. Engine mode (I/O from API files)
   2. Interactive mode (I/O = Keyboard/Screen)

Application areas:
   1. Testing and performance analysis of compilers and computers
   2. Testing of source program analyzers (LOC, complexity, etc.)
   3. Visual demo of the automatic program generation process

Properties of generated benchmark programs:
   1. Program length is expressed in logical lines of code (LLOC).
   2. Generated programs consist of a sequence of functions denoted
      F1(), F2(),...,Fn(), followed by the main program.
   3. All programs contain random expressions and control structures.

The available control structures are:
[1] arithmetic   [2] if           [3] if-else      [4] switch
[5] while        [6] do           [7] for

BenchMaker1 (BM1) is normally called using a command line parameter:
   bm1  project_directory_path    ,  or
   bm1 "project directory path"
Without the project directory path BM1 enters the interactive mode.

The project directory contains the following files:
   1. bm1inpar.txt  (file containing  BM1 input data)
   2. bm1outpar.txt (names and parameters of generated output files)
   3. All generated source C++ program files (one or more)

The bm1inpar.txt file contains (in any order) the name of control
structure followed by the weight of control structure. The weights
are used to compute the relative frequencies of control structures.
All weights must be nonnegative, as in the following example:

   ARITHMETIC  1       |
   IF          1.5     |  Weights can be any
   IF_ELSE     2       |  nonnegative real values.
   SWITCH      0       |  bm1 will automatically
   WHILE       4       |  check and normalize
   DO          2       |  these values (sum = 1).
   FOR         3       |

In addition, we have to specify  the size of generated programs as
the size of individual function (LLOCperFUN) and the sequence of
sizes of generated programs from the minimum (LLOCmin) to the maximum
(LLOCmax) with the step between programs (LLOCstep). E.g. if we want
to create programs with sizes 200, 400, 600,..., 2000 we must include

   LLOCperFUN  100     |  0 or positive
   LLOCmin     200     |  positive and not less than LLOCperFUN
   LLOCmax     2000    |  not less than LLOCmin
   LLOCstep    200     |  any positive value

Conditions for values of input parameters:
   1. All frequencies must be nonnegative
   2. At least one of input frequencies must be positive (any value)
   3. Input data lines can come in any order
   4. LLOCmin > 0
   5. 0 < LLOCperFUN <= LLOCmin <= LLOCmax
   6. If LLOCperFUN = 0 then only the main program is generated
   7. If LLOCmax < LLOCmin it is automatically set equal to LLOCmin
   8. If LLOCstep < 1 it is automatically set to 1
```

USER MANUAL FOR CREATING A BENCHMARK SEQUENCE USING BM1 IN THE SILENT (ENGINE) MODE

1. Create a directory for experiments with BM1 (e.g. testBM1)

2. In the testBM1 directory store a valid BM1 executable (e.g. bm16.exe)

3. For each experiment create in testBM1 a subdirectory (good names are A,B,C,...)

4. Suppose we use the name A. Inside A store the input parameter file bm1inpar.txt
   that contains the desired distribution of control structures and the desired
   size of generated programs expressed in LLOC (logical lines of code) going
   from the minimum value LLOCmin to the maximum value LLOCmax with the step LLOCstep.

   Sample contents of bm1inpar.txt:

```
ARITHMETIC  0
IF          1
IF_ELSE     2
SWITCH      3
WHILE       4
DO          5
FOR         6
LLOCperFUN  50
LLOCmin     200
LLOCmax     1000
LLOCstep    200
```

5. Open a command line window and change directory to testBM1

6. Enter 'bm16 A' to run BM1 in the silent (engine) mode (no displayed results)

7. Inside directory A you will find the sequence of programs that (in the above
   example) contain 200, 400, 600, 800, and 1000 LLOC. BM1 will also create the
   file bm1outpar.txt that contains the following table:
   <program name> <desired size in LLOC> <generated LLOC> <generated PLOC>

   Sample contents of bm1outpar.txt:

```
BM1A1.cpp       200       193       268
BM1A2.cpp       400       405       584
BM1A3.cpp       600       611       880
BM1A4.cpp       800       797      1141
BM1A5.cpp      1000      1020      1477
```

NOTE 1. You may run 'bm16 A' multiple times and each time the content of the A
        directory will be replaced with the new files.

NOTE 2. It is possible to generate the benchmark sequence in interactive mode if we
        run bm16 without command line parameters and then enter the project directory
        path. We do NOT suggest the use of this mode because BM1 will generate excessive
        output. Interactive mode is suitable mostly for generating and investigating
        a single program.

NOTE 3. bm16 always stores the values of generated LLOC and PLOC in the file bm1outpar.txt

Press Return to continue ...

CASE STUDIES

(a) Creating a single program
(b) Creating a sequence of programs

CREATING A SINGLE PROGRAM

```
Project directory path (enter "." for default parameters) = .

Project Directory Path     = .
Project Name               = default
Program Name               = .\BM1default1.cpp
Input Parameter File Name  = bm1inpar.txt
Output Parameter File Name = bm1outpar.txt

Default:  Uniform distribution of control structures
          Generation of a single program .\BM1default1.cpp
          Function size = 40 LLOC
          Program size  = 100 LLOC

Do you want to modify the function or program size (y/n)? y

Function Size (FS>=0) and Program Size (PSmin = 10 LLOC) = 200 10000

Input parameters:

arithmetic = 14.286%
if         = 14.286%
if-else    = 14.286%
switch     = 14.286%
while      = 14.286%
do         = 14.286%
for        = 14.286%
LLOCperFUN =     200
LLOCmin    =   10000
LLOCmax    =   10000
LLOCstep   =       1

Would you like to modify these weights (y/n)? n
```

FUNCTION GENERATION TRACE:

Func	Size	Err[%]	Des_LLOC	Ach_LLOC	Err[%]	Phys_lines	Program Generation Rate
1	218	9.00%	200 D	218 A	9.00%	355 PLOC	6812 LLOC/sec 11094 PLOC/sec
2	199	-0.50%	400 D	417 A	4.25%	673 PLOC	6619 LLOC/sec 10683 PLOC/sec
3	187	-6.50%	600 D	604 A	0.67%	967 PLOC	7744 LLOC/sec 12397 PLOC/sec
4	208	4.00%	800 D	812 A	1.50%	1304 PLOC	7382 LLOC/sec 11855 PLOC/sec
5	194	-3.00%	1000 D	1006 A	0.60%	1615 PLOC	8048 LLOC/sec 12920 PLOC/sec
6	196	-2.00%	1200 D	1202 A	0.17%	1926 PLOC	7656 LLOC/sec 12268 PLOC/sec
7	195	-2.50%	1400 D	1397 A	-0.21%	2236 PLOC	8122 LLOC/sec 13000 PLOC/sec
8	208	4.00%	1600 D	1605 A	0.31%	2561 PLOC	7906 LLOC/sec 12616 PLOC/sec
9	223	11.50%	1800 D	1828 A	1.56%	2924 PLOC	7779 LLOC/sec 12443 PLOC/sec
10	194	-3.00%	2000 D	2022 A	1.10%	3233 PLOC	8088 LLOC/sec 12932 PLOC/sec
11	180	-10.00%	2200 D	2202 A	0.09%	3515 PLOC	7809 LLOC/sec 12465 PLOC/sec
12	212	6.00%	2400 D	2414 A	0.58%	3859 PLOC	8128 LLOC/sec 12993 PLOC/sec
13	182	-9.00%	2600 D	2596 A	-0.15%	4144 PLOC	7915 LLOC/sec 12634 PLOC/sec
14	221	10.50%	2800 D	2817 A	0.61%	4504 PLOC	7825 LLOC/sec 12511 PLOC/sec
15	192	-4.00%	3000 D	3009 A	0.30%	4811 PLOC	8024 LLOC/sec 12829 PLOC/sec
16	188	-6.00%	3200 D	3197 A	-0.09%	5110 PLOC	7855 LLOC/sec 12555 PLOC/sec
17	213	6.50%	3400 D	3410 A	0.29%	5444 PLOC	8081 LLOC/sec 12900 PLOC/sec
18	197	-1.50%	3600 D	3607 A	0.19%	5757 PLOC	7962 LLOC/sec 12709 PLOC/sec
19	227	13.50%	3800 D	3834 A	0.89%	6127 PLOC	7905 LLOC/sec 12633 PLOC/sec
20	178	-11.00%	4000 D	4012 A	0.30%	6405 PLOC	8024 LLOC/sec 12810 PLOC/sec
21	205	2.50%	4200 D	4217 A	0.40%	6733 PLOC	7927 LLOC/sec 12656 PLOC/sec
22	193	-3.50%	4400 D	4410 A	0.23%	7039 PLOC	8062 LLOC/sec 12868 PLOC/sec
23	208	4.00%	4600 D	4618 A	0.39%	7371 PLOC	7990 LLOC/sec 12753 PLOC/sec
24	186	-7.00%	4800 D	4804 A	0.08%	7668 PLOC	8088 LLOC/sec 12909 PLOC/sec
25	215	7.50%	5000 D	5019 A	0.38%	8018 PLOC	8030 LLOC/sec 12829 PLOC/sec
26	183	-8.50%	5200 D	5202 A	0.04%	8304 PLOC	7918 LLOC/sec 12639 PLOC/sec
27	212	6.00%	5400 D	5414 A	0.26%	8649 PLOC	8057 LLOC/sec 12871 PLOC/sec
28	189	-5.50%	5600 D	5603 A	0.05%	8951 PLOC	7970 LLOC/sec 12733 PLOC/sec
29	203	1.50%	5800 D	5806 A	0.10%	9269 PLOC	7899 LLOC/sec 12611 PLOC/sec
30	204	2.00%	6000 D	6010 A	0.17%	9591 PLOC	8013 LLOC/sec 12788 PLOC/sec
31	195	-2.50%	6200 D	6205 A	0.08%	9901 PLOC	7935 LLOC/sec 12661 PLOC/sec
32	208	4.00%	6400 D	6413 A	0.20%	10238 PLOC	8046 LLOC/sec 12846 PLOC/sec
33	183	-8.50%	6600 D	6596 A	-0.06%	10530 PLOC	7966 LLOC/sec 12717 PLOC/sec
34	203	1.50%	6800 D	6799 A	-0.01%	10852 PLOC	8056 LLOC/sec 12858 PLOC/sec
35	208	4.00%	7000 D	7007 A	0.10%	11178 PLOC	8008 LLOC/sec 12775 PLOC/sec
36	203	1.50%	7200 D	7210 A	0.14%	11508 PLOC	7949 LLOC/sec 12688 PLOC/sec
37	186	-7.00%	7400 D	7396 A	-0.05%	11805 PLOC	8022 LLOC/sec 12804 PLOC/sec
38	198	-1.00%	7600 D	7594 A	-0.08%	12118 PLOC	7969 LLOC/sec 12716 PLOC/sec
39	203	1.50%	7800 D	7797 A	-0.04%	12438 PLOC	8046 LLOC/sec 12836 PLOC/sec
40	229	14.50%	8000 D	8026 A	0.33%	12803 PLOC	8026 LLOC/sec 12803 PLOC/sec
41	182	-9.00%	8200 D	8208 A	0.10%	13098 PLOC	7953 LLOC/sec 12692 PLOC/sec
42	194	-3.00%	8400 D	8402 A	0.02%	13405 PLOC	8025 LLOC/sec 12803 PLOC/sec
43	198	-1.00%	8600 D	8600 A	0.00%	13722 PLOC	7978 LLOC/sec 12729 PLOC/sec
44	198	-1.00%	8800 D	8798 A	-0.02%	14035 PLOC	8042 LLOC/sec 12829 PLOC/sec
45	204	2.00%	9000 D	9002 A	0.02%	14356 PLOC	8002 LLOC/sec 12761 PLOC/sec
46	213	6.50%	9200 D	9215 A	0.16%	14701 PLOC	8076 LLOC/sec 12884 PLOC/sec
47	197	-1.50%	9400 D	9412 A	0.13%	15017 PLOC	8031 LLOC/sec 12813 PLOC/sec
48	192	-4.00%	9600 D	9604 A	0.04%	15322 PLOC	8084 LLOC/sec 12897 PLOC/sec
49	198	-1.00%	9800 D	9802 A	0.02%	15633 PLOC	8041 LLOC/sec 12824 PLOC/sec

End of function generation.

Press Return to continue ...

```
RESULTS:

Generated C++ program is stored in file .\BM1default1.cpp
Desired number of logical lines  (LLOC) = 10000
Achieved number of logical lines (LLOC) = 10016
Program size error                      = 0.16%
Total number of physical lines of code  = 15927
Number of physical lines per LLOC       = 1.59
Total consumed processor time           = 1.24 sec
Average program generation rate         = 8110 LLOC/sec
Achieved maximum depth                  = 6
Achieved maximum breadth                = 8
Program name                            = BM1default1.cpp

-----------------------------------------------------------------
Control structure    Count    Dim  Desired prob.  Achieved prob.
-----------------------------------------------------------------
[1]  arithmetic       7630      0        14.29%        14.31%
[2]  if                205    285        14.29%        14.31%
[3]  if-else           206    285        14.29%        14.31%
[4]  switch            206    285        14.29%        14.31%
[5]  while             206    285        14.29%        14.31%
[6]  do                206    285        14.29%        14.24%
[7]  for               205    285        14.29%        14.24%
-----------------------------------------------------------------
Average absolute error = 0.03%
-----------------------------------------------------------------

Depth distribution:
[0] 4.4%  [1] 4.1%  [2] 3.5%  [3] 4.9%  [4] 13.2%  [5] 70.0%  [6] 0.0%

 Achieved (top) and Desired (bottom) Breadth Distributions:
[0] 0.0%  [1] 5.0%  [2] 5.0%  [3] 10.0%  [4] 20.0%  [5] 40.0%  [6] 20.0%  [7] 0.0%
[0] 0.0%  [1] 5.0%  [2] 5.0%  [3] 10.0%  [4] 20.0%  [5] 40.0%  [6] 20.0%  [7] 0.0%

Demo option (R=regular, S=slow, f=fast, F=fastest, X=skip): F


15637  int main(void)
15638  {
15639     int I;
15640     clock_t StartTick = clock();
15641     for(I=0; I<285; I++) IFcnt[I]   =0;
15642     for(I=0; I<285; I++) IFEcnt[I]  =0;
15643     for(I=0; I<285; I++) SWcnt[I]   =0;
15644     for(I=0; I<285; I++) WHILEcnt[I]=0;
15645     for(I=0; I<285; I++) DOcnt[I]   =0;
15646     for(I=0; I<285; I++) FORcnt[I]  =0;
15647     long int sum=0;
15648
15649     sum += F1( ) ;
15650     sum += F2( ) ;
15651     sum += F3( ) ;
15652     sum += F4( ) ;
15653     sum += F5( ) ;
15654     sum += F6( ) ;
15655     sum += F7( ) ;
15656     sum += F8( ) ;
15657     sum += F9( ) ;
15658     sum += F10( ) ;
15659     sum += F11( ) ;
15660     sum += F12( ) ;
15661     sum += F13( ) ;
15662     sum += F14( ) ;
15663     sum += F15( ) ;
15664     sum += F16( ) ;
15665     sum += F17( ) ;
15666     sum += F18( ) ;
15667     sum += F19( ) ;
15668     sum += F20( ) ;
15669     sum += F21( ) ;
15670     sum += F22( ) ;
15671     sum += F23( ) ;
```

```
15672     sum += F24( ) ;
15673     sum += F25( ) ;
15674     sum += F26( ) ;
15675     sum += F27( ) ;
15676     sum += F28( ) ;
15677     sum += F29( ) ;
15678     sum += F30( ) ;
15679     sum += F31( ) ;
15680     sum += F32( ) ;
15681     sum += F33( ) ;
15682     sum += F34( ) ;
15683     sum += F35( ) ;
15684     sum += F36( ) ;
15685     sum += F37( ) ;
15686     sum += F38( ) ;
15687     sum += F39( ) ;
15688     sum += F40( ) ;
15689     sum += F41( ) ;
15690     sum += F42( ) ;
15691     sum += F43( ) ;
15692     sum += F44( ) ;
15693     sum += F45( ) ;
15694     sum += F46( ) ;
15695     sum += F47( ) ;
15696     sum += F48( ) ;
15697     sum += F49( ) ;
15698
15699     {
15700        int a,b,c,d,e,f,g,h,i,j,k,l,m,n;
15701        a=b=c=d=e=f=g=h=i=j=k=l=m=n=1;
15702        l -= (i-d*j*l+n+j+d*l*a)%100;
15703        if( ++IFcnt[204]%10 )
15704        {
15705           if( ++IFEcnt[205]%2 )
15706           {
15707              a += (a-l)%100;
15708              e -= (m+a+n-g*j*e+k*h+n+d*k)%100;
15709              i += (c*b-a)%100;
15710              e -= (n-l+f-a-m+e+m*b+l-m)%100;
15711           }
15712           else
15713           {
15714
15715              switch( ++SWcnt[203]%3 )
15716              {
15717
15718              case 1:
15719              {
15720                 while( ++WHILEcnt[203]%5 )
15721                 {
15722                    i -= (i+i+h-h+j-e+e+l+f*b+g)%100;
15723                    b += (i+g+b-e+n+j-i-d-b-n)%100;
15724                    h -= (i*b-a*b*b+h)%100;
15725                    m += (j+f)%100;
15726                    b -= (b-e-k-a+a*n*c-a-h-a)%100;
15727                 }
15728                 do
15729                 {
15730                    k -= (k*k+d+f*i-a-j+h-k+a-c+j)%100;
15731                    b -= (l+i-k-i-l)%100;
15732                    c += (j+m)%100;
15733                    c += (l+m*i-a)%100;
15734                 } while( ++DOcnt[204]%5 );
15735                 for( ; ++FORcnt[203]%5 ; )
15736                 {
15737                    d -= (c*e*a-b)%100;
15738                    h += (a+l+h+h-l)%100;
15739                    g  = (d*n+d+f-l*l-n+b-k+e-i-d)%100;
15740                    n -= (f*c-l*k*b+g-a+j*m-m+m-h+g)%100;
15741                    f -= (b-l-l*h-j*c+l*i*n-h-i-f+e-f)%100;
15742                    n += (g*h*m+c+k+g+f-a)%100;
15743                 }
15744                 i -= (c-j+b+a+l-h-g+i+g*d-m)%100;
```

```
15745                if( ++IFcnt[202]%10 )
15746                {
15747                   k += (h+j*g-f-i+m-k-n+a+g+a+b-j)%100;
15748                   d += (b+g+j+b-i-m*k*i-l-g+l-l*i)%100;
15749                   b -= (g-l+l*g*b+m+l)%100;
15750                   l -= (h-n-c)%100;
15751                   i += (i+n*c-d*b-c*f-j-f+j+j)%100;
15752                }
15753             }
15754             break;
15755
15756             case 2:
15757             {
15758                if( ++IFEcnt[203]%2 )
15759                {
15760                   n += (a*l)%100;
15761                   j += (g+j-n*i*a-e*e-a+f-l-m)%100;
15762                   i -= (g-h)%100;
15763                   j -= (h*n)%100;
15764                }
15765                else
15766                {
15767                   d += (f-j*n+f+i+f*m-h+f+d)%100;
15768                   n -= (j*d+c)%100;
15769                   k += (l-c*e-i+b)%100;
15770                   i -= (i+k-i+c-g+k-c*g)%100;
15771                   f -= (i+i-m+f-m-j*j+m*d-g-l+d)%100;
15772                }
15773
15774                switch( ++SWcnt[204]%3 )
15775                {
15776
15777                case 1:
15778                {
15779                   l += (m-k+l+d+e+k-n)%100;
15780                   i += (f+n-h+j+l+h+i+n)%100;
15781                   f += (m+j*b*e*k*b+b-j+k*n-b*b+e+i)%100;
15782                   l -= (c+f+m+h-d-g+e+n-a*l-l)%100;
15783                   c += (l-b+a-m+i-n-j+m)%100;
15784                   h += (a+m*f+a+j)%100;
15785                }
15786                break;
15787
15788                case 2:
15789                {
15790                   i -= (j-l*l)%100;
15791                   b -= (c+i-i-a)%100;
15792                   e -= (b-n*a-e*l*h*b+g*d+d+l+a+b+f)%100;
15793                   k -= (m+i+i-i+g-f*g+c)%100;
15794                   d -= (h-m*b*f-j-d+b-l*m+j+i)%100;
15795                }
15796                break;
15797
15798                default:
15799                {
15800                   h -= (g+n+m+i+e-a+g+a+c-m+e*g)%100;
15801                }
15802                }
15803
15804                while( ++WHILEcnt[204]%5 )
15805                {
15806                   d += (l-c+m-m+d*k)%100;
15807                   e -= (d+f*e-f+h-i+d-l)%100;
15808                }
15809             }
15810             break;
15811
15812             default:
15813             {
15814                do
15815                {
15816                   f -= (m+c*i-i-c+c-i*m+g-k+i*j*e*h)%100;
15817                   f += (b*b+d-d-l-g)%100;
```

```
15818                  l += (e-a+l+f-g-j-c-g-a-i-h-m)%100;
15819                  k -= (k-e+g)%100;
15820               } while( ++DOcnt[205]%5 );
15821               for( ; ++FORcnt[204]%5 ; )
15822               {
15823                  m -= (i-h*j+j*k-l-l+e+g-l)%100;
15824                  k -= (h*l+d+c-b)%100;
15825                  g += (i*b+d+a-e-n-n)%100;
15826                  c -= (c*a-k-a-c+i*l)%100;
15827                  k += (i+m+m+n-n+g*d*e-n)%100;
15828                  e -= (l*a-f-m-n*h-j+e+f*b+c+j-c-m)%100;
15829               }
15830               n += (g+i+i+a-g-g-c-e-l-b*l)%100;
15831               if( ++IFcnt[203]%10 )
15832               {
15833                  n += (e+e-d+a-n*j-k-a*e*e)%100;
15834                  a += (c-c-g*c*j-k*f+n+n+d+c-h-d*n)%100;
15835                  d += (c*a+d-b+k+l+b-c+h-j-e)%100;
15836                  h -= (l-m*i*i-k*n-n*l*l-f)%100;
15837                  m += (j+k-g*a*f)%100;
15838               }
15839               if( ++IFEcnt[204]%2 )
15840               {
15841                  d += (b-h*m-k-b)%100;
15842                  e -= (c+l*l+h-i+e+n*c)%100;
15843                  m -= (j-j*l+f+a+m+l+f-g*g+b)%100;
15844                  c += (i-g*d+m+d-h)%100;
15845                  a -= (k-m+m-d*b)%100;
15846               }
15847               else
15848               {
15849                  m -= (h-a*c-f-f-j)%100;
15850                  g -= (n-k*k*e-k*e*f+m*n*e*c)%100;
15851                  c += (c+c)%100;
15852               }
15853            }
15854            }
15855
15856
15857            switch( ++SWcnt[205]%3 )
15858            {
15859
15860            case 1:
15861            {
15862               while( ++WHILEcnt[205]%5 )
15863               {
15864                  h  = (h-k-j+l-n-c+c-g)%100;
15865                  k -= (a+h)%100;
15866                  m += (m+g+m-a)%100;
15867                  f += (d-h-c*b+i)%100;
15868                  g += (n+m*f-f+h+a)%100;
15869                  d += (l*b+g-a-a+l+k)%100;
15870               }
15871               a += (n+g+a+g-d+m*a+c)%100;
15872               j += (l-b-f-e+m+f+i-h+b+f*n+f+c)%100;
15873               j += (l+f+m*a)%100;
15874            }
15875            break;
15876
15877            case 2:
15878            {
15879               h += (i-e-i+g-f+c-n+d*l+k-b+j)%100;
15880               f -= (d*d+k-i-a+j+k+a*k+l*n+j+m-a)%100;
15881               d += (g-b-i-m+f)%100;
15882               k += (a-b+i)%100;
15883               b += (e*b+h-g-l+l-c-d-b)%100;
15884            }
15885            break;
15886
15887            default:
15888            {
15889               m -= (m*b-k-d+n*c-d*k)%100;
15890               h -= (b+i)%100;
```

```
15891                e += (d-g-h+b+i+f-l)%100;
15892                j -= (c-l+j*a+m+h*e+k+f+l-g+b)%100;
15893                a -= (c-b-l+l-i+n-d-f-m)%100;
15894             }
15895             }
15896
15897             c -= (n+k-k-f+e+d+k)%100;
15898             l += (i-g)%100;
15899             m -= (d+i*b+j)%100;
15900             h += (b+a-f*l-g*l-c-k)%100;
15901          }
15902          l += (l-k-a)%100;
15903          c -= (l-g+b*j-l-e)%100;
15904          d -= (b+f*g*e-a-a+f+d+l-h+c)%100;
15905       }
15906       h -= (m-j+h-h-m)%100;
15907       c -= (c-f*n+c)%100;
15908       h += (k+m*h+a*i+b-b+f-b*a)%100;
15909       i += (h-c)%100;
15910       h -= (m+l+j-c*e+j*j)%100;
15911       j -= (b+b)%100;
15912       sum += (a+b+c+d+e+f+g+h+i+j+k+l+m+n)%100 ;
15913    }
15914
15915    cout << "\nChecksum = " << sum;
15916    for(I=sum=0; I<205; I++) sum += IFcnt[I];
15917    cout << "\nIF frequency:      Static = " << 205 << "   Dynamic = " << sum ;
15918    for(I=sum=0; I<206; I++) sum += IFEcnt[I];
15919    cout << "\nIF-ELSE frequency: Static = " << 206 << "   Dynamic = " << sum ;
15920    for(I=sum=0; I<206; I++) sum += SWcnt[I];
15921    cout << "\nSWITCH frequency:  Static = " << 206 << "   Dynamic = " << sum ;
15922    for(I=sum=0; I<206; I++) sum += WHILEcnt[I];
15923    cout << "\nWHILE frequency:   Static = " << 206 << "   Dynamic = " << sum ;
15924    for(I=sum=0; I<206; I++) sum += DOcnt[I];
15925    cout << "\nDO frequency:      Static = " << 206 << "   Dynamic = " << sum ;
15926    for(I=sum=0; I<205; I++) sum += FORcnt[I];
15927    cout << "\nFOR frequency:     Static = " << 205 << "   Dynamic = " << sum ;
15928    cout << "\nRun Time = " << double(clock()-StartTick)/CLOCKS_PER_SEC << " sec\n\n";
15929
15930    return 0;
15931 }
```

```
                                Press Return to continue ...
```

Execution of BM1eefault1 using MS VCPP in debug mode (and DevC++ that generates exactly the same results

```
Checksum = 96
IF frequency:      Static = 205   Dynamic = 11380
IF-ELSE frequency: Static = 206   Dynamic = 10948
SWITCH frequency:  Static = 206   Dynamic = 9368
WHILE frequency:   Static = 206   Dynamic = 28920
DO frequency:      Static = 206   Dynamic = 31405
FOR frequency:     Static = 205   Dynamic = 41045
Run Time = 0.031 sec

Press any key to continue
```

Execution of BM1eefault1 using MS VCPP in release mode

```
Checksum = 96
IF frequency:      Static = 205   Dynamic = 11380
IF-ELSE frequency: Static = 206   Dynamic = 10948
SWITCH frequency:  Static = 206   Dynamic = 9368
WHILE frequency:   Static = 206   Dynamic = 28920
DO frequency:      Static = 206   Dynamic = 31405
FOR frequency:     Static = 205   Dynamic = 41045
Run Time = 0.015 sec

Press any key to continue
```

CREATING A SEQUENCE OF PROGRAMS

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Jozo Dujmovic>cd C:\CentralFiles\Jozo\P R O G R A M
S\BenchMaker1\Release\TestBM16

C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16>dir
 Volume in drive C has no label.
 Volume Serial Number is 849F-4F11

 Directory of C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16

03/12/2014  09:48 AM    <DIR>          .
03/12/2014  09:48 AM    <DIR>          ..
03/12/2014  09:55 AM    <DIR>          A
03/11/2014  11:49 PM           241,664 bm16.exe
               1 File(s)        241,664 bytes
               3 Dir(s)  10,736,291,840 bytes free

C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16>cd A

C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16\A>dir
 Volume in drive C has no label.
 Volume Serial Number is 849F-4F11

 Directory of C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16\A

03/12/2014  09:55 AM    <DIR>          .
03/12/2014  09:55 AM    <DIR>          ..
03/12/2014  09:40 AM               181 bm1inpar.txt
               1 File(s)            181 bytes
               2 Dir(s)  10,736,283,648 bytes free

C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16\A>type bm1inpar.txt
ARITHMETIC  10
IF          30
IF_ELSE     30
SWITCH      30
WHILE       20
DO          15
FOR         30
LLOCperFUN  100
LLOCmin     200
LLOCmax     2000
LLOCstep    100

C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16\A>cd ..
```

```
C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16>bm16 A

C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16>cd A

C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16\A>dir
 Volume in drive C has no label.
 Volume Serial Number is 849F-4F11

 Directory of C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16\A

03/12/2014  09:57 AM    <DIR>          .
03/12/2014  09:57 AM    <DIR>          ..
03/12/2014  09:57 AM             9,015 BM1A1.cpp
03/12/2014  09:57 AM            52,898 BM1A10.cpp
03/12/2014  09:57 AM            57,956 BM1A11.cpp
03/12/2014  09:57 AM            62,077 BM1A12.cpp
03/12/2014  09:57 AM            66,062 BM1A13.cpp
03/12/2014  09:57 AM            70,917 BM1A14.cpp
03/12/2014  09:57 AM            75,726 BM1A15.cpp
03/12/2014  09:57 AM            81,404 BM1A16.cpp
03/12/2014  09:57 AM            85,075 BM1A17.cpp
03/12/2014  09:57 AM            89,001 BM1A18.cpp
03/12/2014  09:57 AM            94,289 BM1A19.cpp
03/12/2014  09:57 AM            15,126 BM1A2.cpp
03/12/2014  09:57 AM            19,396 BM1A3.cpp
03/12/2014  09:57 AM            23,797 BM1A4.cpp
03/12/2014  09:57 AM            28,747 BM1A5.cpp
03/12/2014  09:57 AM            33,361 BM1A6.cpp
03/12/2014  09:57 AM            38,346 BM1A7.cpp
03/12/2014  09:57 AM            43,149 BM1A8.cpp
03/12/2014  09:57 AM            47,884 BM1A9.cpp
03/12/2014  09:40 AM               181 bm1inpar.txt
03/12/2014  09:57 AM               988 bm1outpar.txt
              21 File(s)        995,395 bytes
               2 Dir(s)  10,735,280,128 bytes free

C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16\A>type bm1outpar.txt
           BM1A1.cpp       200       197       286
           BM1A2.cpp       300       318       492
           BM1A3.cpp       400       409       632
           BM1A4.cpp       500       500       776
           BM1A5.cpp       600       602       937
           BM1A6.cpp       700       698      1090
           BM1A7.cpp       800       802      1255
           BM1A8.cpp       900       902      1408
           BM1A9.cpp      1000      1005      1572
          BM1A10.cpp      1100      1114      1742
          BM1A11.cpp      1200      1225      1917
          BM1A12.cpp      1300      1314      2057
          BM1A13.cpp      1400      1408      2207
          BM1A14.cpp      1500      1508      2365
          BM1A15.cpp      1600      1614      2534
          BM1A16.cpp      1700      1734      2731
          BM1A17.cpp      1800      1813      2855
          BM1A18.cpp      1900      1898      2979
          BM1A19.cpp      2000      2006      3156

C:\CentralFiles\Jozo\P R O G R A M S\BenchMaker1\Release\TestBM16\A>
```

Now you can combine the LLOC, PLOC, and memory consumption results to analyze the memory consumption and the density of code