МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Качество и метрология программного обеспечения»

ТЕМА: «Построение операционной графовой модели программы (ОГМП) и расчет характеристик эффективности ее выполнения методом эквивалентных преобразований»

Студент гр. 7304 Моторин Е.В.

Преподаватель Ефремов М.А.

Санкт-Петербург

2021

Цель работы

Построение операционной графовой модели программы (ОГМП) и расчет характеристик эффективности ее выполнения методом эквивалентных преобразований.

Формулировка задания

1.1. Построение ОГМП.

Для рассматривавшегося в лабораторных работах 1-3 индивидуального задания разработать операционную модель управляющего графа программы на основе схемы алгоритма. При выполнении работы рекомендуется для упрощения обработки графа исключить диалог при выполнении операций ввода-вывода данных, а также привести программу к структурированному виду.

Выбрать вариант графа с нагруженными дугами, каждая из которых должна представлять фрагмент программы, соответствующий линейному участку или ветвлению. При расчете вероятностей ветвлений, зависящих от распределения данных, принять равномерное распределение обрабатываемых данных в ограниченном диапазоне (например, [0,100] - для положительных чисел или [-100,100] - для произвольных чисел). В случае
ветвлений, вызванных проверкой выхода из цикла, вероятности рассчитываются исходя априорных сведений о числе повторений цикла. Сложные случаи оценки вероятностей ветвлений согласовать с преподавателем.

В качестве параметров, характеризующих потребление ресурсов, использовать времена выполнения команд соответствующих участков программы. С помощью монитора Sampler выполнить оценку времен выполнения каждого линейного участка в графе программы.

1.2. Расчет характеристик эффективности выполнения программы методом эквивалентных преобразований.

Полученную в части 2.1 данной работы ОГМП, представить в виде графа с нагруженными дугами, у которого в качестве параметров, характеризующих потребление ресурсов на дуге ij, использовать тройку

{Pij,Mij,Dij }, где:

Pij - вероятность выполнения процесса для дуги ij,

Mij - мат.ожидание потребления ресурса процессом для дуги ij, Dij - дисперсия потребления ресурса процессом для дуги ij.

В качестве потребляемого ресурса в данной работе рассматривается время процессора, а оценками мат. ожиданий времен для дуг исходного графа следует принять времена выполнения операторов (команд), соответствующих этим дугам участков программы. Дисперсиям исходных дуг следует присвоить нулевые значения.

Получить описание полученной ОГМП на входном языке пакета CSA III в виде поглощающей марковской цепи (ПМЦ) – (англ.) AMC (absorbing Markov chain) и/или эргодической марковской цепи (ЭМЦ) - EMC (ergodic Markov chain).

С помощью предоставляемого пакетом CSA III меню действий выполнить расчет среднего времени и дисперсии времени выполнения как для всей программы, так и для ее фрагментов, согласованных с преподавателем. Сравнить полученные результаты с результатами измерений, полученными в работе 3.

Ход работы

Построение операционной графовой модели программы

Исходный текст программы приведен в приложении А.

Граф управления программы

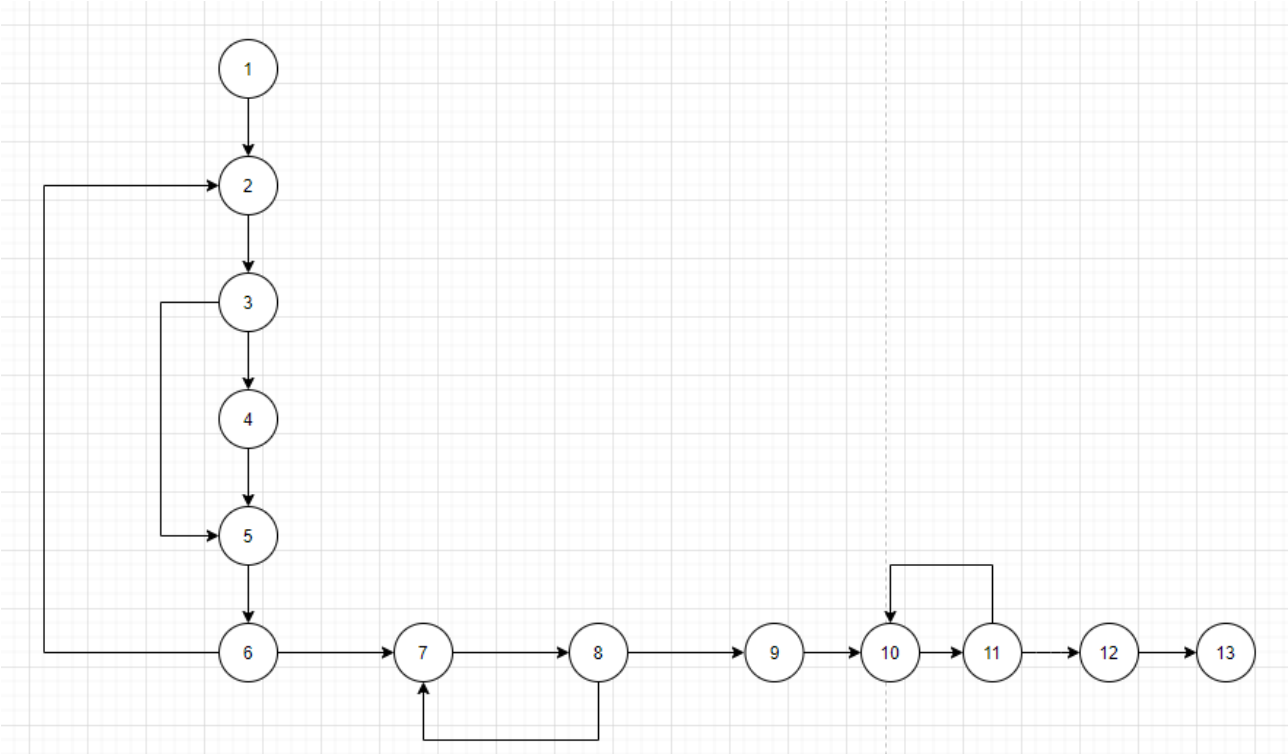


Рисунок 1 – Граф управления программы

Профилирование

Текст программы (подготовленный для профилирования) приведен в приложении Б.

Результаты профилирования

1) X = 5, ordr = 3

Результаты профилирования с измерением времен ФУ:

```
Таблица с результатами измерений ( используется 23 из 416 записей )

-----------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-----------------------------------------------------------------------
 1 :  20  1 :  22               70.40             1              70.40
-----------------------------------------------------------------------
 1 :  22  1 :  30              251.43             1             251.43
-----------------------------------------------------------------------
 1 :  30  1 :  32                0.00             1               0.00
-----------------------------------------------------------------------
 1 :  32  1 :  34                4.19            10               0.42
-----------------------------------------------------------------------
 1 :  34  1 :  40                0.00             1               0.00
 1 :  34  1 :  36                5.03             9               0.56
-----------------------------------------------------------------------
 1 :  36  1 :  38              108.11             9              12.01
-----------------------------------------------------------------------
```

```
 1 :  38  1 :  40              1.68            9              0.19
-------------------------------------------------------------------
 1 :  40  1 :  44            505.37           10             50.54
-------------------------------------------------------------------
 1 :  44  1 :  32            217.07            9             24.12
 1 :  44  1 :  46              9.22            1              9.22
-------------------------------------------------------------------
 1 :  46  1 :  51            105.60            1            105.60
-------------------------------------------------------------------
 1 :  51  1 :  53              0.00            1              0.00
-------------------------------------------------------------------
 1 :  53  1 :  59            767.70           12             63.97
-------------------------------------------------------------------
 1 :  59  1 :  53            201.14           11             18.29
 1 :  59  1 :  61              9.22            1              9.22
-------------------------------------------------------------------
 1 :  61  1 :  64            155.89            1            155.89
-------------------------------------------------------------------
 1 :  64  1 :  71             87.16            1             87.16
-------------------------------------------------------------------
 1 :  71  1 :  79             87.16            1             87.16
-------------------------------------------------------------------
 1 :  79  1 :  83             71.24            1             71.24
-------------------------------------------------------------------
 1 :  83  1 :  86            286.63            1            286.63
-------------------------------------------------------------------
 1 :  86  1 :  90            114.82            2             57.41
-------------------------------------------------------------------
 1 :  90  1 :  86             20.95            1             20.95
 1 :  90  1 :  92             20.95            1             20.95
-------------------------------------------------------------------
 1 :  92  1 :  95              0.84            1              0.84
-------------------------------------------------------------------
 1 :  95  1 :  97              0.00            1              0.00
-------------------------------------------------------------------
```

Суммарное время всех ФУ равно 3101,80 мкс.

Расчет вероятностей и затрат ресурсов для дуг управляющего графа.

	Номера строк	Количество проходов	Вероятность
L_{1-2} = 321.83 мкс	20:22, 22:30, 30:32	1, 1, 1	1.0
L_{2-3} = 0.42 мкс	32:34	10	1.0
L_{3-4} = 12.57 мкс	34:36, 36:38	9, 9	0.9
L_{3-5} = 0.00 мкс	34:40	1	0.1
L_{4-5} = 0.19 мкс	38:40	9	0.9
L_{5-6} = 50.54 мкс	40:44	10	1.0
L_{6-2} = 24.12 мкс	44:32	9	0.9

L_{6-7} = 114.82 мкс	44:46, 46:51, 51:53	1, 1, 1	0.1
L_{7-8} = 63.97 мкс	53:59	12	1.0
L_{8-7} = 18.29 мкс	59:53	11	0.92
L_{8-9} = 9.22 мкс	59:61	1	0.08
L_{9-10} = 688.08 мкс	61:64, 64:71, 71:79, 79:83, 83:86	1, 1, 1, 1, 1	1.0
L_{10-11} = 57.41 мкс	86:90	2	1.0
L_{11-10} = 20.95 мкс	90:86	1	0.5
L_{11-12} = 20.95 мкс	90:92	1	0.5
L_{12-13} = 0.84 мкс	92:95, 95:97	1, 1	1

Операционная графовая модель программы.

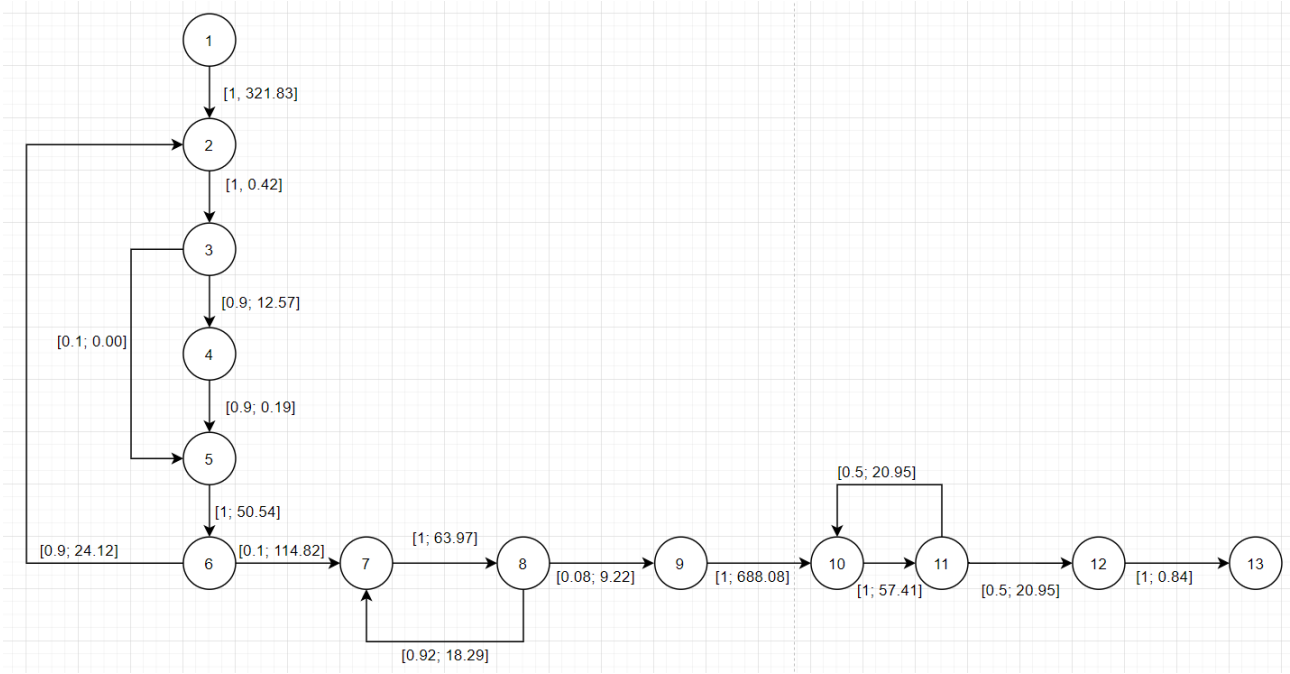
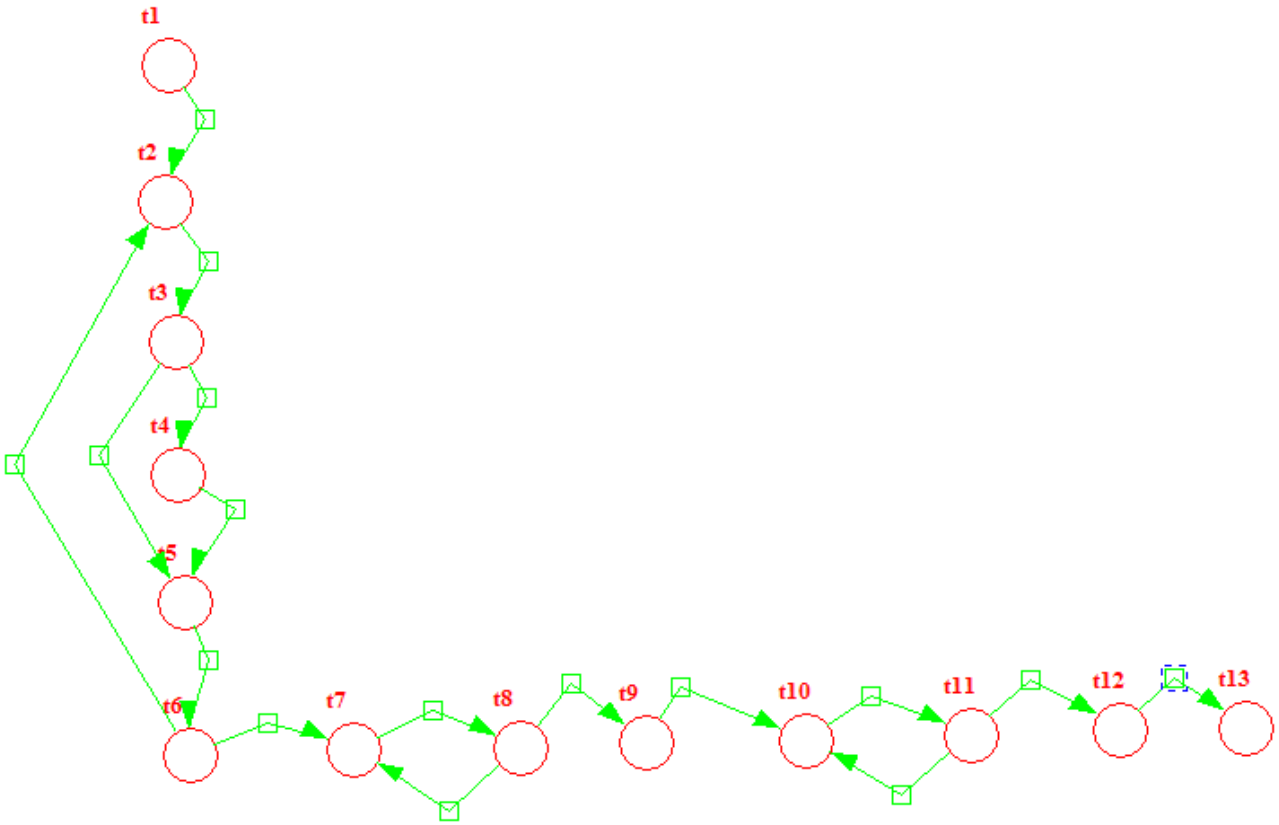


Рисунок 2 – Операционная графовая модель программы

Расчет характеристик эффективности выполнения программы с помощью пакета CSA III методом эквивалентных преобразований ГНД



Описание модели

lr4.xml

```
 <model type = "Objects::AMC::Model" name = "lr4">
    <node type = "Objects::AMC::Top" name = "t1"></node>
    <node type = "Objects::AMC::Top" name = "t2"></node>
    <node type = "Objects::AMC::Top" name = "t3"></node>
    <node type = "Objects::AMC::Top" name = "t4"></node>
    <node type = "Objects::AMC::Top" name = "t5"></node>
    <node type = "Objects::AMC::Top" name = "t6"></node>
    <node type = "Objects::AMC::Top" name = "t7"></node>
    <node type = "Objects::AMC::Top" name = "t8"></node>
    <node type = "Objects::AMC::Top" name = "t9"></node>
    <node type = "Objects::AMC::Top" name = "t10"></node>
    <node type = "Objects::AMC::Top" name = "t11"></node>
    <node type = "Objects::AMC::Top" name = "t12"></node>
    <node type = "Objects::AMC::Top" name = "t13"></node>
    <link type = "Objects::AMC::Link" name = "t1-->t2" probability = "1.0" intensity
= "321.83" deviation = "0.0" source = "t1" dest = "t2"></link>
    <link type = "Objects::AMC::Link" name = "t2-->t3" probability = "1.0" intensity
= "0.42" deviation = "0.0" source = "t2" dest = "t3"></link>
    <link type = "Objects::AMC::Link" name = "t3-->t4" probability = "0.9" intensity
= "12.57" deviation = "0.0" source = "t3" dest = "t4"></link>
```

```
      <link type = "Objects::AMC::Link" name = "t3-->t5" probability = "0.1" intensity
= "0.0" deviation = "0.0" source = "t3" dest = "t5"></link>
      <link type = "Objects::AMC::Link" name = "t4-->t5" probability = "0.9" intensity
= "0.19" deviation = "0.0" source = "t4" dest = "t5"></link>
      <link type = "Objects::AMC::Link" name = "t5-->t6" probability = "1.0" intensity
= "50.54" deviation = "0.0" source = "t5" dest = "t6"></link>
      <link type = "Objects::AMC::Link" name = "t6-->t2" probability = "0.9" intensity
= "24.12" deviation = "0.0" source = "t6" dest = "t2"></link>
      <link type = "Objects::AMC::Link" name = "t6-->t7" probability = "0.1" intensity
= "114.82" deviation = "0.0" source = "t6" dest = "t7"></link>
      <link type = "Objects::AMC::Link" name = "t7-->t8" probability = "1.0" intensity
= "63.97" deviation = "0.0" source = "t7" dest = "t8"></link>
      <link type = "Objects::AMC::Link" name = "t8-->t7" probability = "0.92" intensity
= "18.29" deviation = "0.0" source = "t8" dest = "t7"></link>
      <link type = "Objects::AMC::Link" name = "t8-->t9" probability = "0.08" intensity
= "9.22" deviation = "0.0" source = "t8" dest = "t9"></link>
      <link type = "Objects::AMC::Link" name = "t9-->t10" probability = "1.0" intensity
= "688.08" deviation = "0.0" source = "t9" dest = "t10"></link>
      <link type = "Objects::AMC::Link" name = "t10-->t11" probability = "1.0"
intensity = "57.41" deviation = "0.0" source = "t10" dest = "t11"></link>
      <link type = "Objects::AMC::Link" name = "t11-->t10" probability = "0.5"
intensity = "20.95" deviation = "0.0" source = "t11" dest = "t10"></link>
      <link type = "Objects::AMC::Link" name = "t11-->t12" probability = "0.5"
intensity = "20.95" deviation = "0.0" source = "t11" dest = "t12"></link>
      <link type = "Objects::AMC::Link" name = "t12-->t13" probability = "1.0"
intensity = "0.84" deviation = "0.0" source = "t12" dest = "t13"></link>
    </model>
```

Результаты

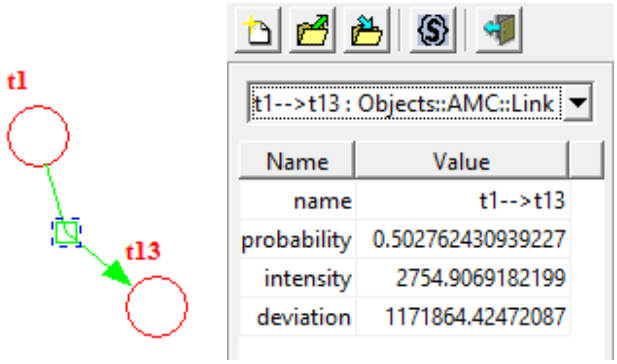


Рисунок 3 – Результаты расчета

Вывод

При выполнении лабораторной работы была построена операционная графовая модель заданной программы, нагрузочные параметры которой были оценены с помощью профилировщика Sampler и методом эквивалентных преобразований с помощью пакета CSA III были вычислены математическое ожидание и дисперсия времени выполнения. Результаты сравнения этих характеристик с полученными в работе 3 согласуются (л.р. 3 – 3111.80 мкс, л.р. 4 – 2754.90 мкс, отличие 12.9%).

ПРИЛОЖЕНИЕ А

Исходный текст программы

```
1. #include <math.h>
2. #include <stdio.h>
3. #include "Sampler.h"
4.
5. float bessy (float x, float n) {
6.       const  float small = 1.0E-8;
7.       const  float euler = 0.57721566;
8.       const  float pi    = 3.1415926;
9.       const  float pi2   = 0.63661977;
10.
11.      int     j;
12.      float  x2,sum,sum2,t,t2,
13.             ts,term,xx,y0,y1,
14.             ya,yb,yc,ans,a,b,
15.             sina,cosa;
16.
17.      if (x<12) {
18.             xx = 0.5 * x;
19.             x2 = xx * xx;
20.             t= log(xx) + euler;
21.             sum = 0.0;
22.             term = t;
23.             y0 = t;
24.             j = 0;
25.             do{
26.                    j=j+1;
27.                    if (j != 1) {sum = sum + 1 / (j - 1);}
28.                    ts = t - sum;
29.                    term = -x2 * term / (j * j)*(1 - 1/(j * ts));
30.                    y0 = y0 + term;
31.             }while ( fabs(term) >= small);
32.             term = xx * (t - 0.5);
33.             sum = 0.0;
34.             y1 = term;
35.             j = 1;
36.             do{
37.                    j = j + 1;
38.                    sum = sum + 1.0 / (j - 1);
39.                    ts = t - sum;
40.                    term = (-x2 * term)/(j * (j - 1))*((ts - 0.5/j)/(ts + 0.5/(j
   - 1)));
41.                    y1 = y1 + term;
42.             }while (fabs(term) >= small);
43.             y0 = pi2 * y0;
44.             y1 = pi2 * (y1 - 1/x);
45.             if (n==0.0) {ans = y0;}
46.             else {
47.                    if (n==1.0) {ans = y1;}
48.                    else
49.                    {
50.                           ts = 2.0 / x;
51.                           ya = y0;
52.                           yb = y1;
53.                           for (j=2;  j<ceil(n+0.01);j+1)
54.                           {
```

```
                                    yc = ts * (j - 1) * yb - ya;
                                    ya = yb;
                                    yb = yc;
                              }
                              ans = yc;
                        }
                  }
                  return ans;
      }
      else    return       sqrt(2 /(pi * x)) * sin(x - pi/4 - n * pi/2);
}

void main (void)
{
      float   x;
      float   ordr;
      int     done;
      float  ans;
      done = 0;
      ordr = 1;
      do{
            if (ordr<0.0) {done = 1;}
            else
            {
                  do{
                        x = 5;
                  }while (x < 0.0);
                  ans = bessy(x,ordr);
                  ordr = -1;
            }
      }while (done == 0)
}
```

ПРИЛОЖЕНИЕ Б

Текст программы, приведенный для профилирования

```
1.  #include <math.h>
2.  #include <stdio.h>
3.  #include "Sampler.h"
4.
5.  float   ordr = 1, x, ans1;
6.  int     done = 0;
7.
8.  float bessy () {
9.          const  float small          = 1.0E-8;
10.         const  float euler          = 0.57721566;
11.         const  float pi    = 3.1415926;
12.         const  float pi2  = 0.63661977;
13.
14.         int     j;
15.         float     x2,sum,sum2,t,t2,
16.                   ts,term,xx,y0,y1,
17.                   ya,yb,yc,ans,a,b,
18.                   sina,cosa;
19.
20.         SAMPLE;
21.         if (x<12) {
22.                   SAMPLE;
23.                   xx = 0.5 * x;
24.                   x2 = xx * xx;
25.                   t= log(xx) + euler;
26.                   sum = 0.0;
27.                   term = t;
28.                   y0 = t;
29.                   j = 0;
30.                   SAMPLE;
31.                   do{
32.                             SAMPLE;
33.                             j+=1;
34.                             SAMPLE;
35.                             if (j != 1) {
36.                                       SAMPLE;
37.                                       sum = sum + 1 / (j - 1);
38.                                       SAMPLE;
39.                             }
40.                             SAMPLE;
41.                             ts = t - sum;
42.                             term = -x2 * term / (j * j)*(1 - 1/(j * ts));
43.                             y0 += term;
44.                             SAMPLE;
45.                   }while ( fabs(term) >= small);
46.                   SAMPLE;
```

```
47.             term = xx * (t - 0.5);
48.             sum = 0.0;
49.             y1 = term;
50.             j = 1;
51.             SAMPLE;
52.             do{
53.                     SAMPLE;
54.                     j = j + 1;
55.                     sum = sum + 1.0 / (j - 1);
56.                     ts = t - sum;
57.                     term = (-x2 * term)/(j * (j - 1))*((ts - 0.5/j)/(ts + 0.5/(j - 1)));
58.                     y1 = y1 + term;
59.                     SAMPLE;
60.             }while (fabs(term) >= small);
61.             SAMPLE;
62.             y0 *= pi2;
63.             y1 = pi2 * (y1 - 1/x);
64.             SAMPLE;
65.             if (ordr==0.0) {
66.                     SAMPLE;
67.                     ans = y0;
68.                     SAMPLE;
69.             }
70.             else {
71.                     SAMPLE;
72.                     if (ordr==1.0) {
73.                             SAMPLE;
74.                             ans = y1;
75.                             SAMPLE;
76.                     }
77.                     else
78.                     {
79.                             SAMPLE;
80.                             ts = 2.0 / x;
81.                             ya = y0;
82.                             yb = y1;
83.                             SAMPLE;
84.                             for (j=2;  j<ceil(ordr+0.01);j+1)
85.                             {
86.                                     SAMPLE;
87.                                     yc = ts * (j - 1) * yb - ya;
88.                                     ya = yb;
89.                                     yb = yc;
90.                                     SAMPLE;
91.                             }
92.                             SAMPLE;
93.                             ans = yc;
94.                     }
95.                     SAMPLE;
96.             }
97.             SAMPLE;
```

```
98.                 return ans;
99.         }
100.                else       return   sqrt(2 /(pi * x)) * sin(x - pi/4 - ordr * pi/2);
101.        }
102.
103.        void main (void)
104.        {
105.                do{
106.                        if (ordr<0.0) {done = 1;}
107.                        else
108.                        {
109.                                do{
110.                                        x = 5;
111.                                }while (x < 0.0);
112.                                ans1 = bessy();
113.                                ordr = -1;
114.                        }
115.                }while (done == 0);
116.        }
```