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 _____ Соколов И.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

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, [0,100.00] - для положительных чисел или [-100.00, 100.00] - для произвольных чисел). В случае ветвлений, вызванных проверкой выхода из цикла, вероятности рассчитываются исходя из 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, полученные с помощью монитора Sampler в процессе выполнения работы №3. Если требуется, оценить с помощью монитора Sampler времена выполнения неучтенных ранее участков программы.

1.2. Расчет характеристик эффективности выполнения программы методом эквивалентных преобразований.

Полученную в части 1.1 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.

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

Это можно сделать двумя способами:

1) Создать с помощью набора стандартных инструментов, предусмотренных в графическом интерфейсе пакета программ CSA III (рисование); выходные операционные графовые модели отображаются на экране, и могут быть сохранены в виде PNG-изображения, а их описание – в виде XML-файла.

2) Описать непосредственно в виде XML-файла.

Подробные сведения по созданию моделей и их последующей обработке для расчета характеристик эффективности выполнения программы приведены в руководстве работы с пакетом CSA III – файл CSA3 Guide.doc.

Полученный XML-файл описания модели для контроля преподавателем следует **преобразовать в более компактную и удобную для понимания форму** с помощью преобразователя csa.xsl, поместив его в каталог csa-model-html-exporter.

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 результаты расчета среднего времени выполнения программы с результатами измерений, полученными в работе 3.

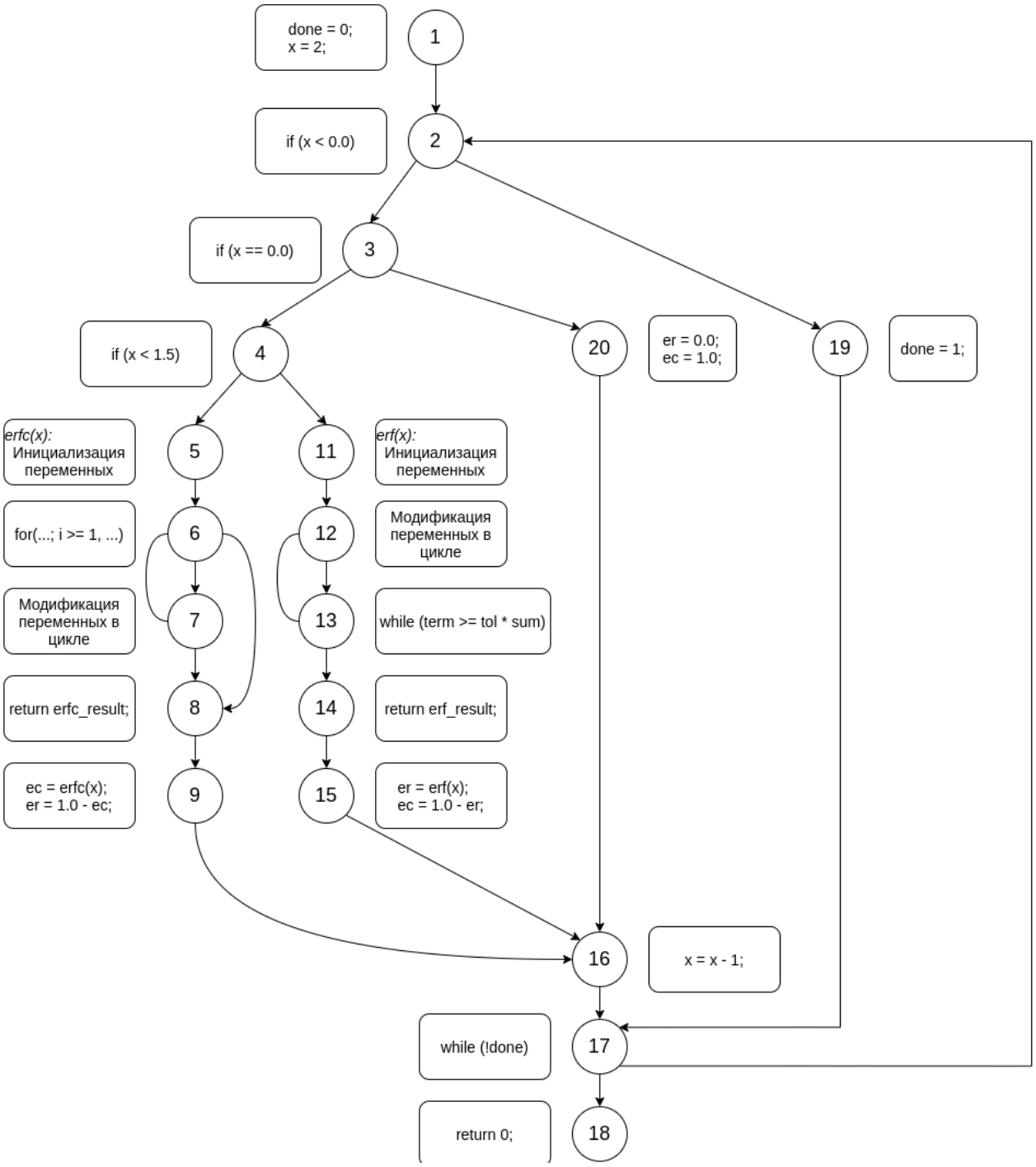
Ход работы.

1. Профилирование программы из ЛР1 (код в приложении А)

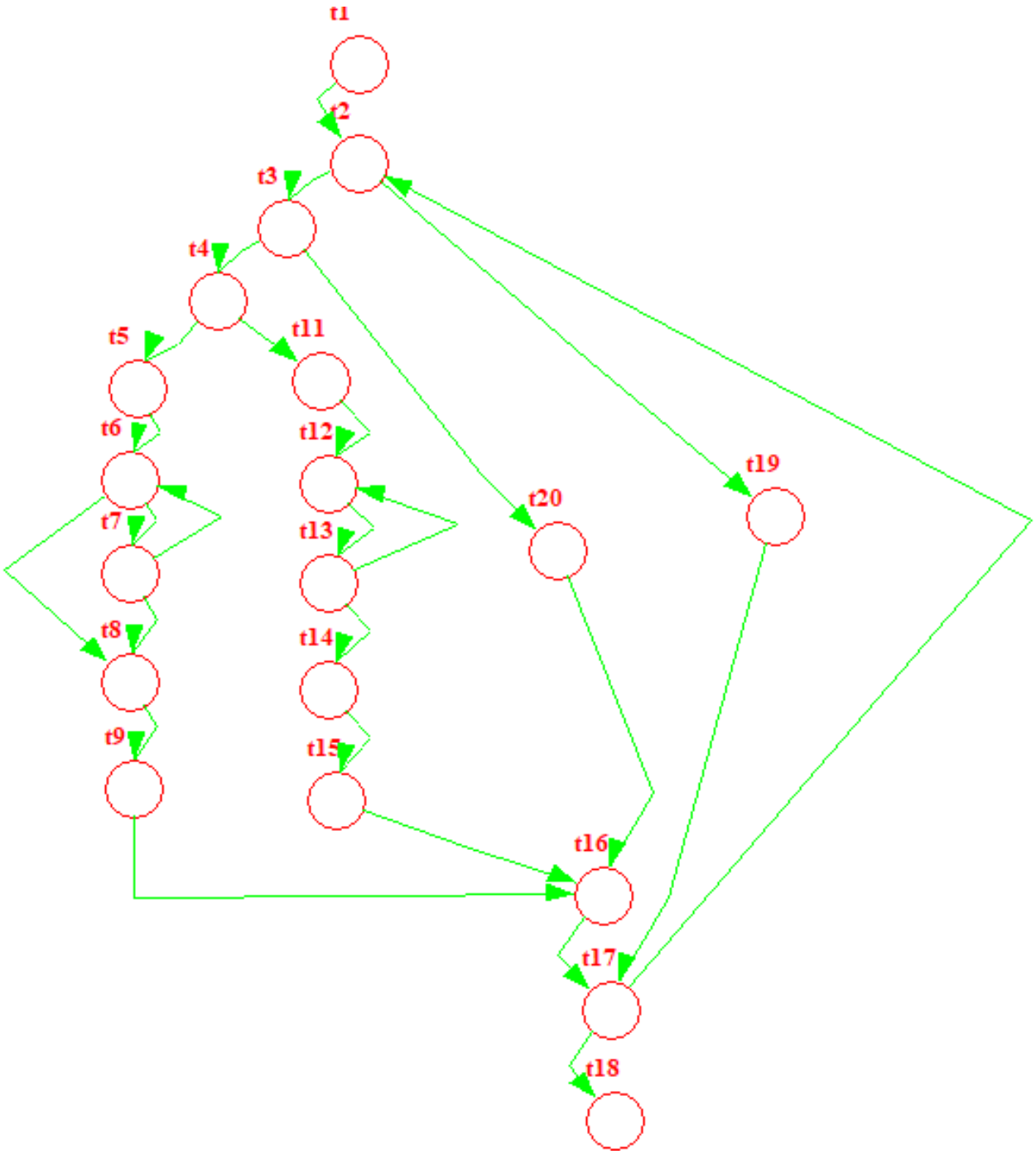
1 : 11 1 : 25	72.91	1	72.91
1 : 25 1 : 30	306.74	7	43.82
1 : 30 1 : 25	118.17	6	19.70
1 : 30 1 : 33	254.78	1	254.78
1 : 33 1 : 91	124.88	1	124.88
1 : 40 1 : 53	297.52	1	297.52
1 : 53 1 : 56	212.04	12	17.67
1 : 56 1 : 53	27.66	11	2.51
1 : 56 1 : 59	234.67	1	234.67
1 : 59 1 : 98	125.71	1	125.71
1 : 65 1 : 69	2.51	1	2.51
1 : 69 1 : 78	101.41	3	33.80
1 : 69 1 : 72	5.87	1	5.87
1 : 72 1 : 74	1.68	1	1.68
1 : 74 1 : 104	3.35	1	3.35
1 : 78 1 : 95	158.40	1	158.40
1 : 78 1 : 88	10.06	1	10.06
1 : 78 1 : 81	5.87	1	5.87
1 : 81 1 : 84	2.51	1	2.51
1 : 84 1 : 101	5.03	1	5.03
1 : 88 1 : 11	54.48	1	54.48
1 : 91 1 : 101	5.87	1	5.87
1 : 95 1 : 40	55.31	1	55.31
1 : 98 1 : 101	88.00	1	88.00
1 : 101 1 : 69	10.06	3	3.35

72.91 + 306.74 + 118.17 + 254.78 + 124.88 + 297.52 + 212.04 + 27.66 + 234.67 + 125.71 + 2.51 + 101.41 + 5.87 + 1.68 + 3.35 + 158.40 + 10.06 + 5.87 + 2.51 + 5.03 + 54.48 + 5.87 + 55.31 + 88.00 + 10.06 = 2285,49 — Суммарное время отдельных замеров

2. Граф программы:



Модель в CSA3:



Расчёт вероятностей и затрат ресурсов для дуг управляющего графа

Дуги	Номера строк	Количество проходов	Время, мкс
11-12	11:25	1	72.91
12-13	25:30	7	306.74 | 43.82
13-12	30:25	6	118.17 | 19.70
13-14	30:33	1	254.78
14-15	33:91	1	124.88
5-6	40:53	1	297.52
6-7	53:56	12	212.04 | 17.67
7-6	56:53	11	27.66 | 2.51

7-8	56:59	1	234.67
8-9	59:98	1	125.71
1-2	65:69	1	2.51
2-3	69:78	3	101.41 | 33.80
2-19	69:72	1	5.87
19-17	72:74	1	1.68
17-18	74:104	1	3.35
3-4-5	78:95	1	158.4
3-4-11	78:88	1	10.06
3-20	78:81	1	5.87
20-16	81:84	1	2.51
16-17	84:101	1	5.03
4-11	88:11	1	54.48
15-16	91:101	1	5.87
4-5	95:40	1	55.31
16-17	98:101	1	88.0
17-2	101:69	3	10.06 | 3.35

По умолчанию подразумевается вероятность перехода равная 1 для всех дуг кроме:

P(2-3) = 0.75

P(2-19) = 0.25

P(3-4) = 0.6666

P(3-20) = 0.3333

P(4-5) = 0.5

P(4-11) = 0.5

P(6-8) = 0

P(7-6) = 11/12 = 0.917

P(7-8) = 1/12 = 0.083

P(13-12) = 6/7 = 0.857

P(13-14) = 1/7 = 0.143

P(17-2) = 0.75

P(17-18) = 0.25

XML файл для CSA3:

```
      <model type = "Objects::AMC::Model" name = "test">
       <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 = "t11"></node>
       <node type = "Objects::AMC::Top" name = "t12"></node>
       <node type = "Objects::AMC::Top" name = "t13"></node>
       <node type = "Objects::AMC::Top" name = "t14"></node>
       <node type = "Objects::AMC::Top" name = "t15"></node>
       <node type = "Objects::AMC::Top" name = "t16"></node>
       <node type = "Objects::AMC::Top" name = "t17"></node>
       <node type = "Objects::AMC::Top" name = "t18"></node>
       <node type = "Objects::AMC::Top" name = "t19"></node>
       <node type = "Objects::AMC::Top" name = "t20"></node>
       <link type = "Objects::AMC::Link" name = "t1-->t2" probability =
"1.0" intensity = "2.51" deviation = "0.0" source = "t1" dest =
"t2"></link>
      <link type = "Objects::AMC::Link" name = "t2-->t3" probability =
"0.75" intensity = "33.8" deviation = "0.0" source = "t2" dest =
"t3"></link>
      <link type = "Objects::AMC::Link" name = "t3-->t4" probability =
"0.6666" intensity = "10.06" deviation = "0.0" source = "t3" dest =
"t4"></link>
      <link type = "Objects::AMC::Link" name = "t5-->t6" probability =
"1.0" intensity = "297.52" deviation = "0.0" source = "t5" dest =
"t6"></link>
      <link type = "Objects::AMC::Link" name = "t4-->t5" probability =
"0.5" intensity = "55.31" deviation = "0.0" source = "t4" dest =
"t5"></link>
      <link type = "Objects::AMC::Link" name = "t4-->t11" probability =
"0.5" intensity = "54.58" deviation = "0.0" source = "t4" dest =
"t11"></link>
```

```
      <link type = "Objects::AMC::Link" name = "t6-->t7" probability =
"1.0" intensity = "17.67" deviation = "0.0" source = "t6" dest =
"t7"></link>
      <link type = "Objects::AMC::Link" name = "t7-->t8" probability =
"0.083" intensity = "234.67" deviation = "0.0" source = "t7" dest =
"t8"></link>
      <link type = "Objects::AMC::Link" name = "t8-->t9" probability =
"1.0" intensity = "125.71" deviation = "0.0" source = "t8" dest =
"t9"></link>
      <link type = "Objects::AMC::Link" name = "t11-->t12" probability =
"1.0" intensity = "72.91" deviation = "0.0" source = "t11" dest =
"t12"></link>
      <link type = "Objects::AMC::Link" name = "t12-->t13" probability =
"1.0" intensity = "43.82" deviation = "0.0" source = "t12" dest =
"t13"></link>
      <link type = "Objects::AMC::Link" name = "t14-->t15" probability =
"1.0" intensity = "124.88" deviation = "0.0" source = "t14" dest =
"t15"></link>
      <link type = "Objects::AMC::Link" name = "t13-->t14" probability =
"0.143" intensity = "254.78" deviation = "0.0" source = "t13" dest =
"t14"></link>
      <link type = "Objects::AMC::Link" name = "t2-->t19" probability =
"0.25" intensity = "5.87" deviation = "0.0" source = "t2" dest =
"t19"></link>
      <link type = "Objects::AMC::Link" name = "t3-->t20" probability =
"0.3333" intensity = "5.87" deviation = "0.0" source = "t3" dest =
"t20"></link>
      <link type = "Objects::AMC::Link" name = "t20-->t16" probability =
"1.0" intensity = "2.51" deviation = "0.0" source = "t20" dest =
"t16"></link>
      <link type = "Objects::AMC::Link" name = "t16-->t17" probability =
"1.0" intensity = "88.0" deviation = "0.0" source = "t16" dest =
"t17"></link>
      <link type = "Objects::AMC::Link" name = "t17-->t18" probability =
"0.25" intensity = "3.35" deviation = "0.0" source = "t17" dest =
"t18"></link>
      <link type = "Objects::AMC::Link" name = "t9-->t16" probability =
"1.0" intensity = "0.0" deviation = "0.0" source = "t9" dest =
"t16"></link>
```

```
      <link type = "Objects::AMC::Link" name = "t15-->t16" probability =
"1.0" intensity = "5.87" deviation = "0.0" source = "t15" dest =
"t16"></link>
      <link type = "Objects::AMC::Link" name = "t6-->t8" probability =
"0.0" intensity = "0.0" deviation = "0.0" source = "t6" dest =
"t8"></link>
      <link type = "Objects::AMC::Link" name = "t7-->t6" probability =
"0.917" intensity = "2.51" deviation = "0.0" source = "t7" dest =
"t6"></link>
      <link type = "Objects::AMC::Link" name = "t13-->t12" probability =
"0.857" intensity = "19.7" deviation = "0.0" source = "t13" dest =
"t12"></link>
      <link type = "Objects::AMC::Link" name = "t17-->t2" probability =
"0.75" intensity = "3.35" deviation = "0.0" source = "t17" dest =
"t2"></link>
      <link type = "Objects::AMC::Link" name = "t19-->t17" probability =
"1.0" intensity = "1.68" deviation = "0.0" source = "t19" dest =
"t17"></link>
    </model>
```

Результат работы программы CSA3:

name	t1-->t18
probability	0.999700067484816
intensity	2308.13299994025
deviation	5234711.28327942

Разность времени Sampler'а (2285.49) с CSA3 (2308.132) ~ 0.9%. Вероятность близка к 1, что свидетельствует о корректном распределении вероятностей по дугам.

Выводы

При выполнении лабораторной работы была построена операционная графовая модель программы из ЛР1, было оценено время выполнения программы с помощью профилировщика Sampler и методом эквивалентных преобразований с помощью пакета CSA III. Результаты сравнения этих характеристик показали, что метод эквивалентных преобразований даёт результаты очень близкие к результатам работы программы Sampler.

ПРИЛОЖЕНИЕ А

Исходный код Лабораторной работы 1 для измерения времени работы отдельных участков кода

```
1.  #include "math.h"
2.  #include "sampler.h"
3.
4.  float x, er, ec;
5.  unsigned char done;
6.
7.  float erf(float x)
8.   /* infinite series expansion of the Gaussian error function */
9.
10. {
11.  SAMPLE;
12.  static const float sqrtpi = 1.7724538;
13.  static const float tol = 1.0E-4;
14.
15.  float x2, sum, sum1, term;
16.  int i;
17.
18.
19.  float erf_result;
20.  x2 = x * x;
21.  sum = x;
22.  term = x;
23.  i = 0;
24.  do {
25.   SAMPLE;
26.   i = i + 1;
27.   sum1 = sum;
28.   term = 2.0 * term * x2 / (1.0 + 2.0 * i);
29.   sum = term + sum1;
30.   SAMPLE;
31.  } while (term >= tol * sum);
32.  erf_result = 2.0 * sum * exp(-x2) / sqrtpi;
33.  SAMPLE;
34.  return erf_result;
35. }    /* erf */
```

```
36.
37.         float erfc(float x)
38.          /* complement of error function */
39.         {
40.          SAMPLE;
41.          static const float sqrtpi = 1.7724538;
42.          int terms = 12;
43.
44.          float x2, u, v, sum;
45.          int i;
46.
47.          float erfc_result;
48.          x2 = x * x;
49.          v = 1.0 / (2.0 * x2);
50.          u = 1.0 + v * (terms + 1.0);
51.          for( i = terms; i >= 1; i --)
52.          {
53.            SAMPLE;
54.            sum = 1.0 + i * v / u;
55.            u = sum;
56.            SAMPLE;
57.          }
58.          erfc_result = exp(-x2) / (x * sum * sqrtpi);
59.          SAMPLE;
60.          return erfc_result;
61.         }      /* ercf */
62.
63.         int main()
64.         {       /* main */
65.          SAMPLE;
66.          done = 0;
67.          x = 2;
68.          do {
69.            SAMPLE;
70.            if (x < 0.0)
71.            {
72.              SAMPLE;
73.               done = 1;
```

```
74.             SAMPLE;
75.           }
76.           else
77.           {
78.             SAMPLE;
79.             if (x == 0.0)
80.             {
81.               SAMPLE;
82.               er = 0.0;
83.               ec = 1.0;
84.               SAMPLE;
85.             }
86.             else if (x < 1.5)
87.             {
88.               SAMPLE;
89.               er = erf(x);
90.               ec = 1.0 - er;
91.               SAMPLE;
92.             }
93.             else
94.             {
95.               SAMPLE;
96.               ec = erfc(x);
97.               er = 1.0 - ec;
98.               SAMPLE;
99.             }  /* if */
100.            x = x - 1;
101.            SAMPLE;
102.          }   /* if */
103.        } while (!done);
104.        SAMPLE;
105.        return 0;
106.      }
```

ПРИЛОЖЕНИЕ Б

Модифицированный код Лабораторной работы 1 для измерения времени работы всей программы

```
#include "math.h"
#include "sampler.h"

float x, er, ec;
unsigned char done;
float sqrtpi = 1.7724538;

float erf(float x)
 /* infinite series expansion of the Gaussian error function */

{
 float x2, sum, term;
 int i;


 float erf_result;
 x2 = x * x;
 sum = x;
 term = x;
 i = 0;
 do {
  i = i + 1;
  term = 2.0 * term * x2 / (1.0 + 2.0 * i);
  sum = term + sum;
 } while (term >= 1.0E-4 * sum);
 erf_result = 2.0 * sum * exp(-x2) / sqrtpi;
 return erf_result;
}    /* erf */

float erfc(float x)
 /* complement of error function */
```

```
{
  float u, v;
  int i;

  float erfc_result;
  v = 1.0 / (2.0 * x * x);
  u = 1.0 + v * 13;
  for( i = 12; i >= 1; i --)
  {
    u = 1.0 + i * v / u;
  }
  erfc_result = exp(-x2) / (x * u * sqrtpi);
  return erfc_result;
}      /* ercf */

int main()
{       /* main */
  SAMPLE;
  er = 0.0;
  ec = 1.0;
  for (x = 1; x <= 2.0; x++) {
    if (x < 1.5) {
      er = erf(x);
      ec = 1.0 - er;
    }
    else {
      ec = erfc(x);
      er = 1.0 - ec;
    }
  }
  SAMPLE;
  return 0;
}
```