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. 6304 Рыбин А.С.

Преподаватель Кирьянчиков В.А.

Санкт-Петербург
2020

Цель

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

Задание

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.

2.2. 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

Выполняется **вариант 14**

Построение операционной графовой модели программы

3. Текст программы (исходный).

Текст программы так же представлен в приложении А (**LINFIT.CPP**).

```
01: #include <stdlib.h>
02:
03: #define SIZE 1000
04:
05: void linfit(const float* x, const float* y, float* y_calc, float* a, float*
b, int n) {
06:     float sum_x, sum_y, sum_xy, sum_x2, xi, yi, sxx, sxy;
07:
08:     sum_x = 0.0;
09:     sum_y = 0.0;
10:     sum_xy = 0.0;
11:     sum_x2 = 0.0;
12:
13:     int i = 0;
14:
15:     for (i = 0; i < n; i++) {
16:         xi = x[i];
17:         yi = y[i];
18:
19:         sum_x += xi;
20:         sum_y += yi;
21:
22:         sum_xy += xi * yi;
23:
24:         sum_x2 += xi * xi;
25:     }
26:
27:     sxx = sum_x2 - sum_x * sum_x / n;
28:     sxy = sum_xy - sum_x * sum_y / n;
29:
30:     *a = sxy / sxx;
31:     *b = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
32:
33:     for (i = 0; i < n; i++) {
```

```
34:         y_calc[i] = *a + *b * x[i];
35:     }
36: }
37:
38: int main(int argc, char* argv[]) {
39:     float x[SIZE];
40:     float y[SIZE];
41:     float y_calc[SIZE];
42:
43:     float a, b;
44:
45:     for (int i = 0; i < SIZE; i++) {
46:         x[i] = rand() % 200 - 100; // [-100, 100)
47:         y[i] = rand() % 200 - 100; // [-100, 100)
48:     }
49:
50:     linfit(x, y, y_calc, &a, &b, SIZE);
51:
52:     return 0;
53: }
54:
```

Граф управления программы

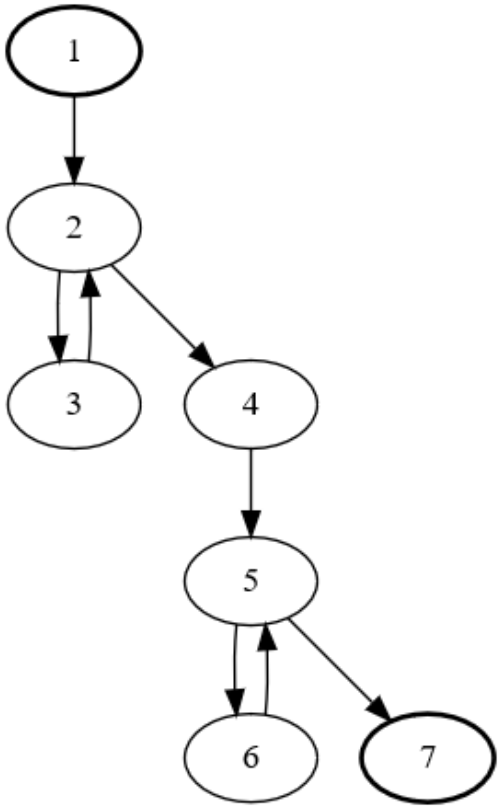

Рисунок 1 – Граф управления программы

4. Профилирование

Текст программы (подготовленный для профилирования).

Текст программы так же представлен в приложении Б (**LINFIT_FUN_PROFILER.CPP**).

```
001: #include <stdlib.h>
002:
003: #include <SAMPLER.H>
004:
005: #define SIZE 1000
006:
007: void linfit(const float* x, const float* y, float* y_calc, float* a, float*
b, int n) {
008:     SAMPLE;
009:
010:     // Функциональный участок №1
011:     // -------------------------
012:
013:     float sum_x, sum_y, sum_xy, sum_x2, xi, yi, sxx, sxy;
014:
015:     sum_x = 0.0;
016:     sum_y = 0.0;
017:     sum_xy = 0.0;
018:     sum_x2 = 0.0;
019:
020:     int i = 0; // Счётчик для цикла
021:
022:     // -------------------------
023:     // Функциональный участок №1
024:
025:     SAMPLE;
026:
027:     // Функциональный участок №2
028:     // -------------------------
029:     //
030:     // i < n
031:     //
032:     // -------------------------
033:     // Функциональный участок №2
034:
```

```
035:     for ( ; i < n ; ) {
036:         SAMPLE;
037:
038:         // Функциональный участок №3
039:         // -------------------------
040:
041:         xi = x[i];
042:         yi = y[i];
043:
044:         sum_x += xi;
045:         sum_y += yi;
046:
047:         sum_xy += xi * yi;
048:
049:         sum_x2 += xi * xi;
050:
051:         i++; // Чтобы померять i < n отдельно
052:
053:         // -------------------------
054:         // Функциональный участок №3
055:
056:         SAMPLE;
057:     }
058:
059:     SAMPLE;
060:
061:     // Функциональный участок №4
062:     // -------------------------
063:
064:     sxx = sum_x2 - sum_x * sum_x / n;
065:     sxy = sum_xy - sum_x * sum_y / n;
066:
067:     *a = sxy / sxx;
068:     *b = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
069:
070:     i = 0; // Для следующего цикла
071:
072:     // -------------------------
073:     // Функциональный участок №4
074:
075:     SAMPLE;
```

```
076:
077:     // Функциональный участок №5
078:     // -------------------------
079:     //
080:     // i < n
081:     //
082:     // -------------------------
083:     // Функциональный участок №5
084:
085:     for ( ; i < n ; ) {
086:         SAMPLE;
087:
088:         // Функциональный участок №6
089:         // -------------------------
090:
091:         y_calc[i] = *a + *b * x[i];
092:
093:         i++; // Чтобы померять i < n отдельно
094:
095:         // -------------------------
096:         // Функциональный участок №6
097:
098:         SAMPLE;
099:     }
100:
101:     // Функциональный участок №7
102:     // -------------------------
103:     // Конец программы
104:
105:     SAMPLE;
106: }
107:
108: int main(int argc, char* argv[]) {
109:     float x[SIZE];
110:     float y[SIZE];
111:     float y_calc[SIZE];
112:
113:     float a, b;
114:
115:     SAMPLE;
116:
```

```
117:     for (int i = 0; i < SIZE; i++) {
118:
119:         SAMPLE;
120:
121:         x[i] = rand() % 200 - 100; // [-100, 100)
122:         y[i] = rand() % 200 - 100; // [-100, 100)
123:
124:         SAMPLE;
125:     }
126:
127:     SAMPLE;
128:
129:     linfit(x, y, y_calc, &a, &b, SIZE);
130:
131:     SAMPLE;
132:
133:     return 0;
134: }
135:
```

Результаты профилирования

```
      Отчет о результатах измерений для программы LI05EC~1.EXE.

      Создан программой Sampler ( версия от Feb 15 1999 )
            1995-98 (c) СПбГЭТУ, Мойсейчук Леонид.



            Список обработанных файлов.

-------------------------------------------------------------------------
   NN            Имя обработанного файла
-------------------------------------------------------------------------
   1.  LI05EC~1.CPP
-------------------------------------------------------------------------


   Таблица с результатами измерений ( используется 14 из 416 записей )

-------------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-------------------------------------------------------------------------
```

```
 1 :   8  1 :  25              4.19            1              4.19
-------------------------------------------------------------------
 1 :  25  1 :  36              1.68            1              1.68
-------------------------------------------------------------------
 1 :  36  1 :  56          18115.46         1000             18.12
-------------------------------------------------------------------
 1 :  56  1 :  36           1420.57          999              1.42
 1 :  56  1 :  59              0.84            1              0.84
-------------------------------------------------------------------
 1 :  59  1 :  75            469.33            1            469.33
-------------------------------------------------------------------
 1 :  75  1 :  86              1.68            1              1.68
-------------------------------------------------------------------
 1 :  86  1 :  98           8533.50         1000              8.53
-------------------------------------------------------------------
 1 :  98  1 :  86           1428.12          999              1.43
 1 :  98  1 : 105              1.68            1              1.68
-------------------------------------------------------------------
 1 : 105  1 : 131              2.51            1              2.51
-------------------------------------------------------------------
 1 : 115  1 : 119              1.68            1              1.68
-------------------------------------------------------------------
 1 : 119  1 : 124          30876.31         1000             30.88
-------------------------------------------------------------------
 1 : 124  1 : 119           1769.22          999              1.77
 1 : 124  1 : 127              1.68            1              1.68
-------------------------------------------------------------------
 1 : 127  1 :   8              7.54            1              7.54
-------------------------------------------------------------------
```

5. Расчёт вероятностей и затрат ресурсов для дуг управляющего графа

Таблица 1 – Вероятности и затраты ресурсов для дуг управляющего графа

	Ресурсы, мкс	**Номера строк**	**Количество проходов**
L1	4.19	8:25	1
L2	1.68, 1.42, 0.84	25:36, 56:36, 56:59	1, 999, 1
L3	18. 12	36:56	1000

L4	469	59:75	1
L5	1.68, 1.43, 1.68	75:86, 98:86, 98:105	1, 999, 1
L6	8.53	86:98	1000
L7	-	-	-

6. Операционная графовая модель программы

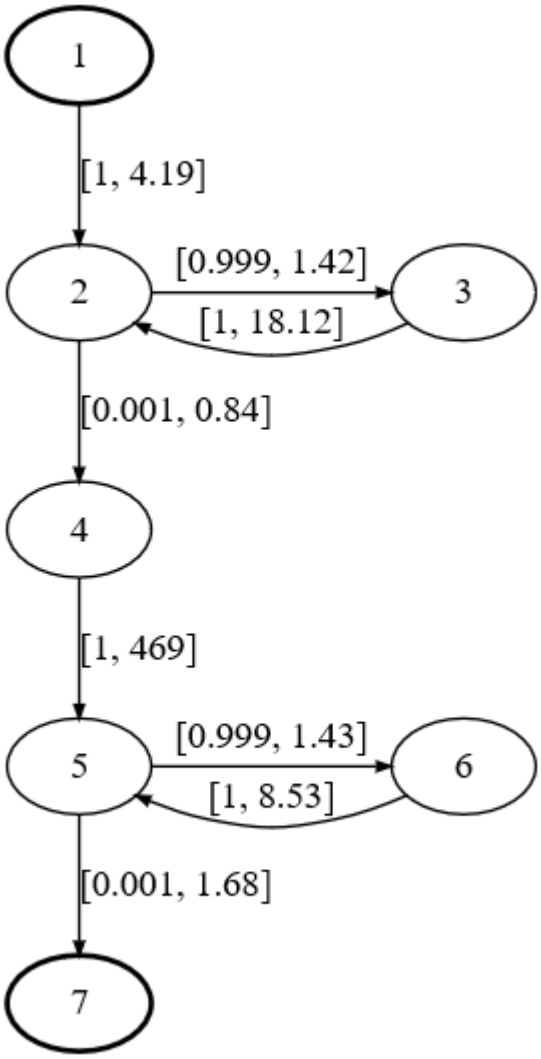

Рисунок 2 – Операционная графовая модель программы

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

ГНД

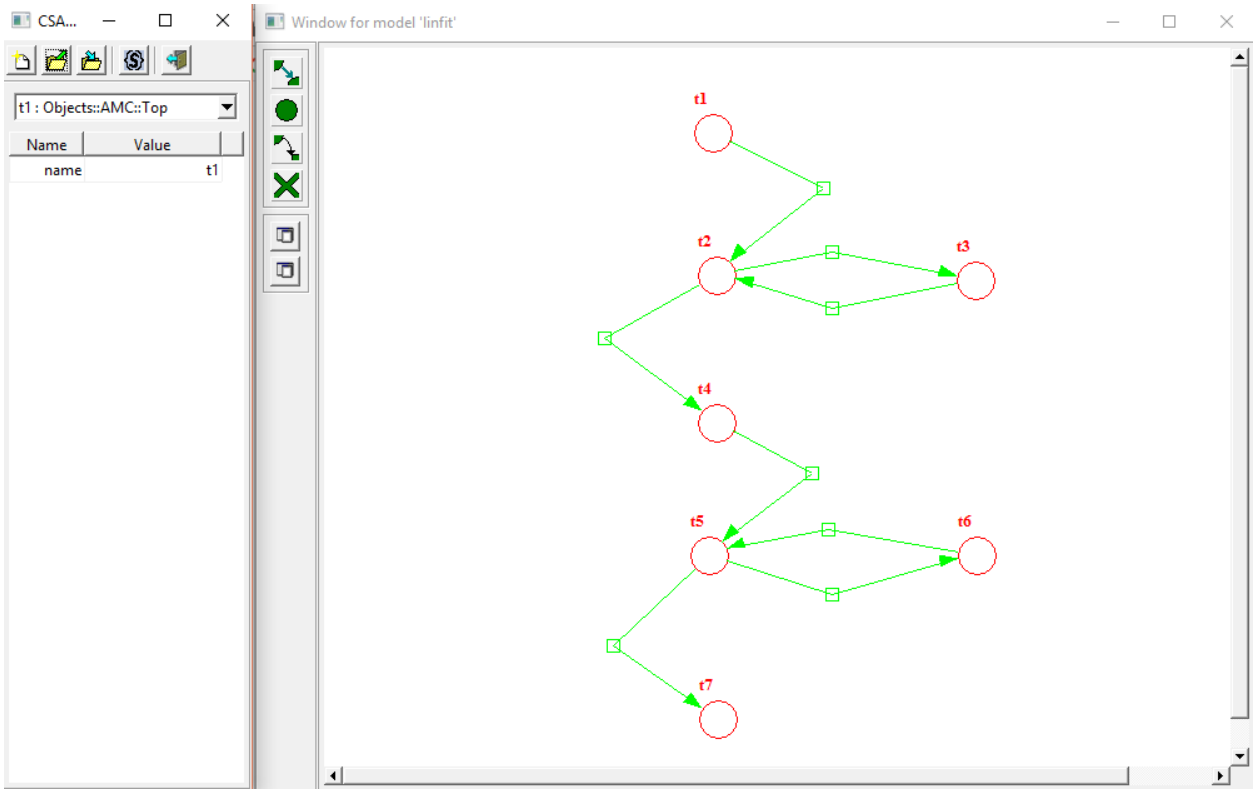

Рисунок 3 – ГНД

7. Описание модели

Описание модели так же приведено в приложении В (**MODEL.XML**).

Model.xml

```
<model type="Objects::AMC::Model" name="linfit">
    <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>

    <link type="Objects::AMC::Link" name="t1-t2" probability="1"
intensity="4.19" deviation="0.0"
                                    source="t1" dest="t2"></link>
```

```
        <link    type="Objects::AMC::Link"    name="t2-t3"    probability="0.999"
intensity="1.42" deviation="0.0"
                                        source="t2" dest="t3"></link>

        <link     type="Objects::AMC::Link"     name="t3-t2"     probability="1"
intensity="18.12" deviation="0.0"
                                        source="t3" dest="t2"></link>

        <link    type="Objects::AMC::Link"    name="t2-t4"    probability="0.001"
intensity="0.84" deviation="0.0"
                                        source="t2" dest="t4"></link>

        <link     type="Objects::AMC::Link"     name="t4-t5"     probability="1"
intensity="469" deviation="0.0"
                                        source="t4" dest="t5"></link>

        <link    type="Objects::AMC::Link"    name="t5-t6"    probability="0.999"
intensity="1.43" deviation="0.0"
                                        source="t5" dest="t6"></link>

        <link     type="Objects::AMC::Link"     name="t6-t5"     probability="1"
intensity="8.53" deviation="0.0"
                                        source="t6" dest="t5"></link>

        <link    type="Objects::AMC::Link"    name="t5-t7"    probability="0.001"
intensity="1.68" deviation="0.0"
                                        source="t5" dest="t7"></link>

    </model>
```

Результаты

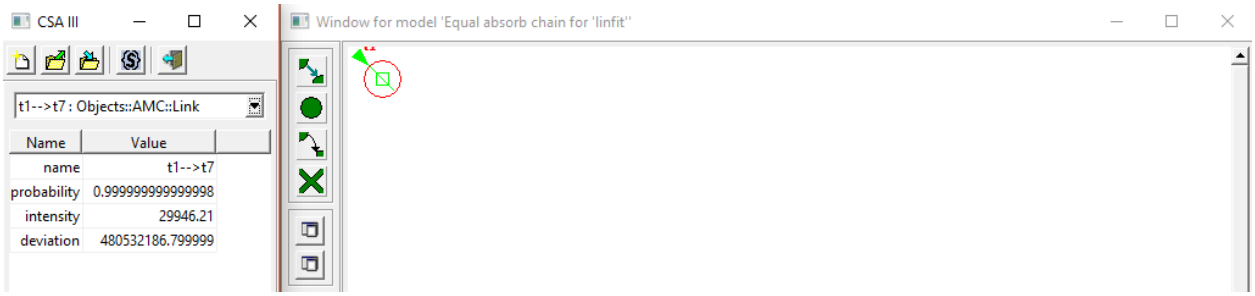

Рисунок 4 – Результаты

Результат отличается от полученного с помощью ПИМ на **0.1%**.

Выводы

В ходе выполнения лабораторной работы было изучено понятие ОГМП программы. Для этого был построен граф управления программы из лабораторной работы №1 и выполнено профилирование её функциональных участков с помощью ПИМ SAMPLER. После был произведён расчёт вероятностей и затрат ресурсов для дуг управляющего графа по результатам профилирования. В результате был получен ОГМП программы.

В ходе моделирования ОГМП средствами программного пакета CSA III было рассчитано время выполнения программы, которое отличается от полученного ПИМ на **0.1%**, что говорит о адекватности построенной модели.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД LINFIT.CPP

```
01: #include <stdlib.h>
02:
03: #define SIZE 1000
04:
05: void linfit(const float* x, const float* y, float* y_calc, float* a, float* b,
int n) {
06:     float sum_x, sum_y, sum_xy, sum_x2, xi, yi, sxx, sxy;
07:
08:     sum_x = 0.0;
09:     sum_y = 0.0;
10:     sum_xy = 0.0;
11:     sum_x2 = 0.0;
12:
13:     int i = 0;
14:
15:     for (i = 0; i < n; i++) {
16:         xi = x[i];
17:         yi = y[i];
18:
19:         sum_x += xi;
20:         sum_y += yi;
21:
22:         sum_xy += xi * yi;
23:
24:         sum_x2 += xi * xi;
25:     }
26:
27:     sxx = sum_x2 - sum_x * sum_x / n;
28:     sxy = sum_xy - sum_x * sum_y / n;
29:
30:     *a = sxy / sxx;
31:     *b = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
32:
```

```
33:     for (i = 0; i < n; i++) {
34:         y_calc[i] = *a + *b * x[i];
35:     }
36: }
37:
38: int main(int argc, char* argv[]) {
39:     float x[SIZE];
40:     float y[SIZE];
41:     float y_calc[SIZE];
42:
43:     float a, b;
44:
45:     for (int i = 0; i < SIZE; i++) {
46:         x[i] = rand() % 200 - 100; // [-100, 100)
47:         y[i] = rand() % 200 - 100; // [-100, 100)
48:     }
49:
50:     linfit(x, y, y_calc, &a, &b, SIZE);
51:
52:     return 0;
53: }
54:
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД LINFIT_FUN_PROFILER.CPP

```
001: #include <stdlib.h>
002:
003: #include <SAMPLER.H>
004:
005: #define SIZE 1000
006:
007: void linfit(const float* x, const float* y, float* y_calc, float* a, float* b,
int n) {
008:     SAMPLE;
009:
010:     // Функциональный участок №1
011:     // -------------------------
012:
013:     float sum_x, sum_y, sum_xy, sum_x2, xi, yi, sxx, sxy;
014:
015:     sum_x = 0.0;
016:     sum_y = 0.0;
017:     sum_xy = 0.0;
018:     sum_x2 = 0.0;
019:
020:     int i = 0; // Счётчик для цикла
021:
022:     // -------------------------
023:     // Функциональный участок №1
024:
025:     SAMPLE;
026:
027:     // Функциональный участок №2
028:     // -------------------------
029:     //
030:     // i < n
031:     //
032:     // -------------------------
```

```
033:     // Функциональный участок №2
034:
035:     for ( ; i < n ; ) {
036:         SAMPLE;
037:
038:         // Функциональный участок №3
039:         // -------------------------
040:
041:         xi = x[i];
042:         yi = y[i];
043:
044:         sum_x += xi;
045:         sum_y += yi;
046:
047:         sum_xy += xi * yi;
048:
049:         sum_x2 += xi * xi;
050:
051:         i++; // Чтобы померять i < n отдельно
052:
053:         // -------------------------
054:         // Функциональный участок №3
055:
056:         SAMPLE;
057:     }
058:
059:     SAMPLE;
060:
061:     // Функциональный участок №4
062:     // -------------------------
063:
064:     sxx = sum_x2 - sum_x * sum_x / n;
065:     sxy = sum_xy - sum_x * sum_y / n;
066:
067:     *a = sxy / sxx;
```

```
068:     *b = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
069:
070:     i = 0; // Для следующего цикла
071:
072:     // -------------------------
073:     // Функциональный участок №4
074:
075:     SAMPLE;
076:
077:     // Функциональный участок №5
078:     // -------------------------
079:     //
080:     // i < n
081:     //
082:     // -------------------------
083:     // Функциональный участок №5
084:
085:     for ( ; i < n ; ) {
086:         SAMPLE;
087:
088:         // Функциональный участок №6
089:         // -------------------------
090:
091:         y_calc[i] = *a + *b * x[i];
092:
093:         i++; // Чтобы померять i < n отдельно
094:
095:         // -------------------------
096:         // Функциональный участок №6
097:
098:         SAMPLE;
099:     }
100:
101:     // Функциональный участок №7
102:     // -------------------------
```

```
103:     // Конец программы
104:
105:     SAMPLE;
106: }
107:
108: int main(int argc, char* argv[]) {
109:     float x[SIZE];
110:     float y[SIZE];
111:     float y_calc[SIZE];
112:
113:     float a, b;
114:
115:     SAMPLE;
116:
117:     for (int i = 0; i < SIZE; i++) {
118:
119:         SAMPLE;
120:
121:         x[i] = rand() % 200 - 100; // [-100, 100)
122:         y[i] = rand() % 200 - 100; // [-100, 100)
123:
124:         SAMPLE;
125:     }
126:
127:     SAMPLE;
128:
129:     linfit(x, y, y_calc, &a, &b, SIZE);
130:
131:     SAMPLE;
132:
133:     return 0;
134: }
135:
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ MODEL.XML

```
<model type="Objects::AMC::Model" name="linfit">
    <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>

    <link      type="Objects::AMC::Link"      name="t1-t2"      probability="1"
intensity="4.19" deviation="0.0"
                                        source="t1" dest="t2"></link>

    <link     type="Objects::AMC::Link"     name="t2-t3"     probability="0.999"
intensity="1.42" deviation="0.0"
                                        source="t2" dest="t3"></link>

    <link      type="Objects::AMC::Link"      name="t3-t2"      probability="1"
intensity="18.12" deviation="0.0"
                                        source="t3" dest="t2"></link>

    <link     type="Objects::AMC::Link"     name="t2-t4"     probability="0.001"
intensity="0.84" deviation="0.0"
                                        source="t2" dest="t4"></link>

    <link      type="Objects::AMC::Link"      name="t4-t5"      probability="1"
intensity="469" deviation="0.0"
                                        source="t4" dest="t5"></link>

    <link     type="Objects::AMC::Link"     name="t5-t6"     probability="0.999"
intensity="1.43" deviation="0.0"
                                        source="t5" dest="t6"></link>
```

```
          <link      type="Objects::AMC::Link"      name="t6-t5"      probability="1"
intensity="8.53" deviation="0.0"
                                          source="t6" dest="t5"></link>

          <link     type="Objects::AMC::Link"     name="t5-t7"     probability="0.001"
intensity="1.68" deviation="0.0"
                                          source="t5" dest="t7"></link>
      </model>
```