

**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. 8304

Сергеев А.Д.

Преподаватель

Кириянчиков В. А.

Санкт-Петербург

2022

### **Цель работы.**

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

### **Задание.**

Для задания из лабораторных работ 1-3 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.

Полученную ОГМП, представить в виде графа с нагруженными дугами, у которого в качестве параметров, характеризующих потребление ресурсов на дуге  $ij$ , использовать тройку  $\{P_{ij}, M_{ij}, D_{ij}\}$ , где:

$P_{ij}$  - вероятность выполнения процесса для дуги  $ij$ ;

$M_{ij}$  - мат. ожидание потребления ресурса процессом для дуги  $ij$ ;

$D_{ij}$  - 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.**

#### **1. Текст программы (исходный).**

Исходный код программы из лабораторных работ 1 - 3 представлен в приложении А.

#### **2. Профилирование.**

Код программы для профилирования, разделенной на функциональные участки, представлен в приложении Б.

#### **3. Граф управления программы.**

Граф управления был построен на основе программы, разбитой на функциональные участки, так как подсчёт затрат времени в лабораторной работе №3 осуществлялся на каждом из таких участков отдельно. На графе также отмечены метки (номера строк) и количество проходов для удобного вычисления вероятностей. Граф представлен на рисунке 1.

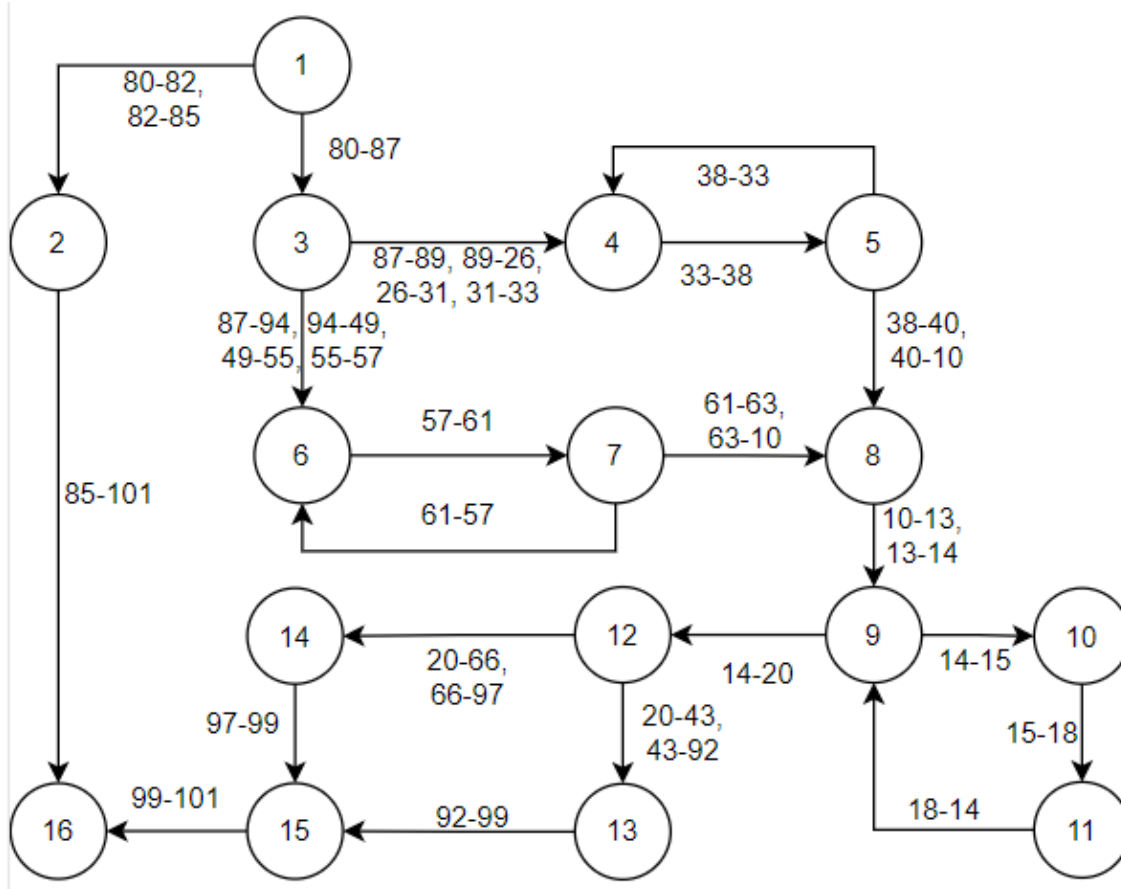


Рисунок 1 - Граф управления

#### 4. Результаты профилирования.

Результаты профилирования из лабораторной работы №3 представлены в таблице 1.

Таблица 1 – Результаты профилирования

исх	прием	общее время	кол-во проходов	среднее время
80	87	57.500	1	57.500
87	94	90.200	1	90.200
94	49	131.700	1	131.700
49	55	38.200	1	38.200
55	57	29.600	1	29.600
57	61	261.200	12	21.767
61	57	144.700	11	13.155
61	63	44.200	1	44.200
63	10	31.200	1	31.200
10	13	26.600	1	26.600
13	14	34.000	1	34.000
14	15	61.300	1	61.300
14	20	47.700	1	47.700
15	18	32.400	1	32.400
18	14	30.000	1	30.000
20	66	38.100	1	38.100
66	97	97.900	1	97.900
97	99	31.600	1	31.600
99	101	23.200	1	23.200

## 5. Расчет вероятностей и затрат ресурсов для дуг управляющего графа.

Результаты расчета представлены в таблице 1.

Таблица 2 – Расчет вероятностей и затрат ресурсов для дуг.

	Номера строк	Количество проходов	Вероятность
$L_{1-2} = 0$	80-82, 82-85	0	0
$L_{1-3} = 57,5$	80-87	1	1
$L_{2-16} = 0$	85-101	0	0
$L_{3-4} = 0$	87-89, 89-26, 26-31, 31-33	0	0
$L_{4-5} = 0$	33-38	0	0
$L_{5-4} = 0$	38-33	0	0
$L_{3-6} = 289,7$	87-94, 94-49, 49-55, 55-57	1	1
$L_{5-8} = 0$	38-40, 40-10	0	0
$L_{6-7} = 21,767$	57-61	12	1
$L_{7-8} = 102$	61-63, 63-10	1	0.0833
$L_{7-6} = 13,155$	61-57	11	0.9167
$L_{8-9} = 60,6$	10-13, 13-14	1	1
$L_{9-10} = 61,3$	14-15	1	0,5
$L_{9-12} = 47,7$	14-20	1	0,5
$L_{10-11} = 32,4$	15-18	1	1
$L_{11-9} = 30$	18-14	1	1
$L_{12-14} = 136$	20-66, 66-97	1	1
$L_{12-13} = 0$	20-43, 43-92	0	0
$L_{13-15} = 0$	92-99	0	0
$L_{14-15} = 31,6$	97-99	1	1
$L_{15-16} = 23,2$	99-101	1	1

## 6. Операционная графовая модель программы.

Операционная графовая модель программы представлена на рисунке 3.

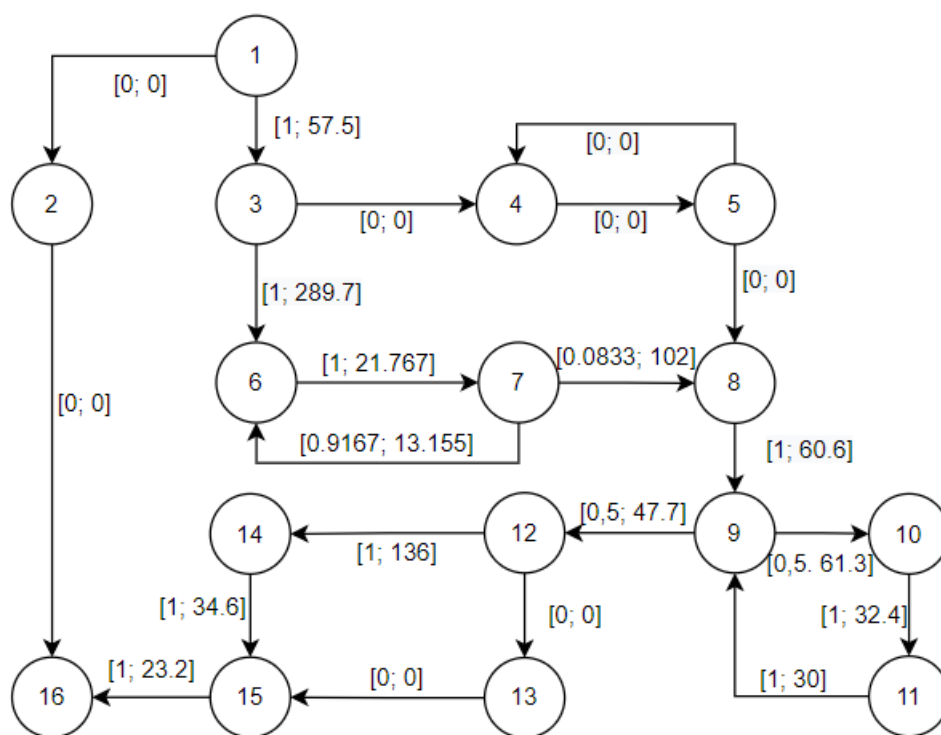


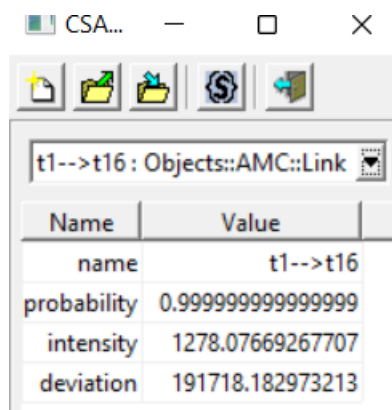
Рисунок 2 - Операционная модель

## 7. Описание модели model.xml.

Описание модели представлено в приложении В.

## 8. Результаты.

Результаты работы программы представлены на рисунке 4.



The screenshot shows a window titled 'CSA...' with a toolbar containing icons for file operations and a settings icon. Below the toolbar is a table with the following data:

Name	Value
name	t1-->t16
probability	0.999999999999999
intensity	1278.07669267707
deviation	191718.182973213

Рисунок 4 - Результаты работы программы

Вероятность равна 1. Согласно расчётам программы, среднее время выполнения программы составляет 1278 мкс. В пункте 4 данного отчёта приведен результат профилирования программы с использованием SAMPLER\_v2, в котором суммарное время выполнения составило 1251,3 мкс. Разница между результатами составляет менее 3%.

### Выводы.

В ходе выполнения лабораторной работы была построена операционная графовая модель заданной программы, нагрузочные параметры которой были оценены с помощью монитора SAMPLER\_v2.



## ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <math.h>
#include <stdio.h>

const double sqrtpi = 1.7724538;
const double tol = 1.0E-4;
const int terms = 12;

// infinite series expansion of the Gaussian error function
double erf (double x) {
    double x2 = x * x;
    double sum = x;
    double term = x;
    int i = 0;
    do {
        i = i + 1;
        double sum1 = sum;
        term = 2.0 * term * x2 / (1.0 + 2.0 * i);
        sum = term + sum1;
    } while (term < tol * sum);
    return 2.0 * sum * exp(-x2) / sqrtpi;
}

// complement of error function
double erfc (double x) {
    double x2,u,v,sum;
    x2 = x * x;
    v = 1.0 / (2.0 * x2);
    u = 1.0 + v * (terms + 1.0);
    int i = terms;
    do {
        sum = 1.0 + i * v / u;
        u = sum;
        i--;
    } while (i >= 1);
    return exp(-x2) / (x * sum * sqrtpi);
}

// evaluation of the gaussian error function
int main () {
    double x, er, ec;
    int done = 1;
    do {
        printf("Arg? ");
        scanf("%lf", &x);
        if (x < 0.0) done = 0;
        else {
            if (x == 0.0) {
                er = 0.0;
                ec = 1.0;
            } else {
                if (x < 1.5) {
                    er = erf(x);
                    ec = 1.0 - er;
                } else {
                    ec = erfc(x);
                    er = 1.0 - ec;
                }
            }
            printf("X = %.8lf; Erf = %.12lf; Erfc = %.12lf\n", x, er, ec);
        }
    }
}
```

```
    } while (done);  
}
```

## ПРИЛОЖЕНИЕ Б.

### КОД ПРОГРАММЫ, РАЗДЕЛЕННОЙ НА ФУНКЦИОНАЛЬНЫЕ УЧАСТКИ

```
#include <math.h>

#include <stdio.h>

#include "sampler.h"

const double sqrtpi = 1.7724538;

const double tol = 1.0E-4;

const int terms = 12;

double my_exp( const double x ){

    SAMPLE;

    double dVal, dTemp;

    int nStep = 1;

    SAMPLE;

    for( dVal = 1.0, dTemp = 1.0; SAMPLE, dTemp >= 1e-6 ; ++nStep ){

        SAMPLE;

        dTemp *= x/nStep;

        dVal += dTemp;

        SAMPLE;

    }

    SAMPLE;

    return dVal;

}

// infinite series expansion of the Gaussian error function

double erf (double x) {

    SAMPLE;

    double x2 = x * x;

    double sum = x;

    double term = x;

    int i = 0;

    SAMPLE;

    do {
```

```

        SAMPLE;

        i = i + 1;

        double sum1 = sum;

        term = 2.0 * term * x2 / (1.0 + 2.0 * i);

        sum = term + sum1;

        SAMPLE;
    } while (term < tol * sum);
    SAMPLE;

    double res = 2.0 * sum * my_exp(-x2) / sqrt(pi);
    SAMPLE;

    return res;
}

// complement of error function
double erfc (double x) {
    SAMPLE;

    double x2,u,v,sum;

    x2 = x * x;

    v = 1.0 / (2.0 * x2);

    u = 1.0 + v * (terms + 1.0);

    int i = terms;

    SAMPLE;

    do {

        SAMPLE;

        sum = 1.0 + i * v / u;

        u = sum;

        i--;

        SAMPLE;

    } while (i >= 1);

    SAMPLE;

    double res = my_exp(-x2) / (x * sum * sqrt(pi));
    SAMPLE;

    return res;
}

```

```

}

// evaluation of the gaussian error function
int main (int argc, char **argv) {
    sampler_init(&argc, argv);

    double x, er, ec;

    int done = 1;

    do {
        printf("Arg? ");
        scanf("%lf", &x);
        if (x < 0.0) done = 0;
        else {
            SAMPLE;

            if (x == 0.0) {
                SAMPLE;

                er = 0.0;
                ec = 1.0;

                SAMPLE;
            } else {
                SAMPLE;

                if (x < 1.5) {
                    SAMPLE;

                    er = erf(x);
                    ec = 1.0 - er;

                    SAMPLE;
                } else {
                    SAMPLE;

                    ec = erfc(x);
                    er = 1.0 - ec;

                    SAMPLE;
                }

                SAMPLE;
            }

            SAMPLE;

            printf("X = %.8lf; Erf = %.12lf; Erfc = %.12lf\n", x, er, ec);
        }
    }
}

```

```
    }  
    } while (done);  
}
```

## ПРИЛОЖЕНИЕ В.

### ОПИСАНИЕ МОДЕЛИ .XML

```
<model type = "Objects::AMC::Model" name = "model">
<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 = "t12"></node>
<node type = "Objects::AMC::Top" name = "t11"></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>

<link type = "Objects::AMC::Link" name = "t1-->t2" probability = "0.0" intensity =
    "0.0" deviation = "0.0" source = "t1" dest = "t2"></link>
<link type = "Objects::AMC::Link" name = "t1-->t3" probability = "1.0" intensity =
    "57.5" deviation = "0.0" source = "t1" dest = "t3"></link>
<link type = "Objects::AMC::Link" name = "t2-->t16" probability = "0.0" intensity =
    "0.0" deviation = "0.0" source = "t2" dest = "t16"></link>
<link type = "Objects::AMC::Link" name = "t3-->t4" probability = "0.0" intensity =
    "0.0" deviation = "0.0" source = "t3" dest = "t4"></link>
<link type = "Objects::AMC::Link" name = "t4-->t5" probability = "0.0" intensity =
    "0.0" deviation = "0.0" source = "t4" dest = "t5"></link>
<link type = "Objects::AMC::Link" name = "t5-->t4" probability = "0.0" intensity =
    "0.0" deviation = "0.0" source = "t5" dest = "t4"></link>
<link type = "Objects::AMC::Link" name = "t3-->t6" probability = "1.0" intensity =
    "289.7" deviation = "0.0" source = "t3" dest = "t6"></link>
<link type = "Objects::AMC::Link" name = "t5-->t8" probability = "0.0" intensity =
    "0.0" deviation = "0.0" source = "t5" dest = "t8"></link>
<link type = "Objects::AMC::Link" name = "t6-->t7" probability = "1.0" intensity =
    "21.767" deviation = "0.0" source = "t6" dest = "t7"></link>
<link type = "Objects::AMC::Link" name = "t7-->t8" probability = "0.0833" intensity =
    "102.0" deviation = "0.0" source = "t7" dest = "t8"></link>
<link type = "Objects::AMC::Link" name = "t7-->t6" probability = "0.9167" intensity =
    "13.155" deviation = "0.0" source = "t7" dest = "t6"></link>
<link type = "Objects::AMC::Link" name = "t8-->t9" probability = "1.0" intensity =
    "60.6" deviation = "0.0" source = "t8" dest = "t9"></link>
<link type = "Objects::AMC::Link" name = "t9-->t10" probability = "0.5" intensity =
    "61.3" deviation = "0.0" source = "t9" dest = "t10"></link>
<link type = "Objects::AMC::Link" name = "t9-->t12" probability = "0.5" intensity =
    "47.7" deviation = "0.0" source = "t9" dest = "t12"></link>
<link type = "Objects::AMC::Link" name = "t10-->t11" probability = "1.0" intensity =
    "32.4" deviation = "0.0" source = "t10" dest = "t11"></link>
<link type = "Objects::AMC::Link" name = "t11-->t9" probability = "1.0" intensity =
    "30.0" deviation = "0.0" source = "t11" dest = "t9"></link>
<link type = "Objects::AMC::Link" name = "t12-->t14" probability = "1.0" intensity =
    "136.0" deviation = "0.0" source = "t12" dest = "t14"></link>
<link type = "Objects::AMC::Link" name = "t12-->t13" probability = "0.0" intensity =
    "0.0" deviation = "0.0" source = "t12" dest = "t13"></link>
<link type = "Objects::AMC::Link" name = "t13-->t15" probability = "0.0" intensity =
    "0.0" deviation = "0.0" source = "t13" dest = "t15"></link>
<link type = "Objects::AMC::Link" name = "t14-->t15" probability = "1.0" intensity =
    "31.6" deviation = "0.0" source = "t14" dest = "t15"></link>
<link type = "Objects::AMC::Link" name = "t15-->t16" probability = "1.0" intensity =
```

```
"23.2" deviation = "0.0" source = "t15" dest = "t16"></link>  
</model>
```