

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МОЭВМ**

**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

\_\_\_\_\_

Мешков М.А.

Преподаватель

\_\_\_\_\_

Ефремов М.А.

Санкт-Петербург

2022

## **Цель работы.**

### **1. Построение ОГМП.**

Для рассматривавшегося в лабораторных работах 1-3 индивидуального задания разработать операционную модель управляющего графа программы на основе схемы алгоритма.

Выбрать вариант графа с нагруженными дугами, каждая из которых должна представлять фрагмент программы, соответствующий линейному участку или ветвлению.

В качестве параметров, характеризующих потребление ресурсов, использовать времена выполнения команд соответствующих участков программы, полученные с помощью монитора Sampler.

**2. Расчет характеристик эффективности выполнения программы методом эквивалентных преобразований.**

Полученную в части 1 данной работы ОГМП, представить в виде графа с нагруженными дугами, у которого в качестве параметров, характеризующих потребление ресурсов на дуге  $ij$ , использовать тройку  $\{ P_{ij}, M_{ij} \}$ , где:

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

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

Выполнить описание построенной ОГМП на входном языке пакета CSA III в виде марковской цепи. 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.

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 выполнения.

1. Для программы из лаб. работы 3 был составлен граф управления — см. рис. 1. Номера вершин соответствуют комментариям в исходном коде (см. в приложении А ).

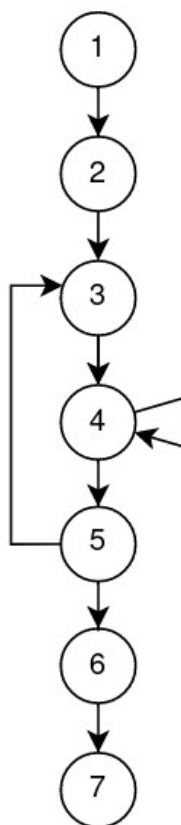


Рисунок 1 - Граф управления программы

2. Программа из лаб. работы 3 была разбита на линейные участки и были расставлены на их границах контрольные точки (КТ) для выполнения с помощью ПИМ SAMPLER измерений и получения профиля выполнения программы, представляющего времена выполнения и количество выполнений каждого ФУ. Исходный код программы с контрольными точками см. в приложении А.

Под управлением SAMPLER была выполнена программа из лаб. Работы 3 — см. табл 1, программа sampler-repeat вызывалась с параметрами 10000 5.

Таблица 1 — Результаты профилирования программы из лаб. работы 3

исх	прием	общее время	кол-во проходов	среднее время	std
41	10	31.444	1	31.444	41.729

исх	прием	общее время	кол-во проходов	среднее время	std
10	21	65.517	1	65.517	18.152
21	27	72.337	7	10.334	64.480
27	30	129.405	7	18.486	49.068
30	30	4163.469	247	16.856	91.118
30	34	190.161	7	27.166	54.001
34	21	67.143	6	11.190	50.890
34	36	26.253	1	26.253	8.156
36	48	39.774	1	39.774	35.069

Также результаты профилирования были преобразованы в вид графа с помощью программы fmdot — см. рис. 2. Видно что структурно этот граф похож на граф управления программы на рис. 1.

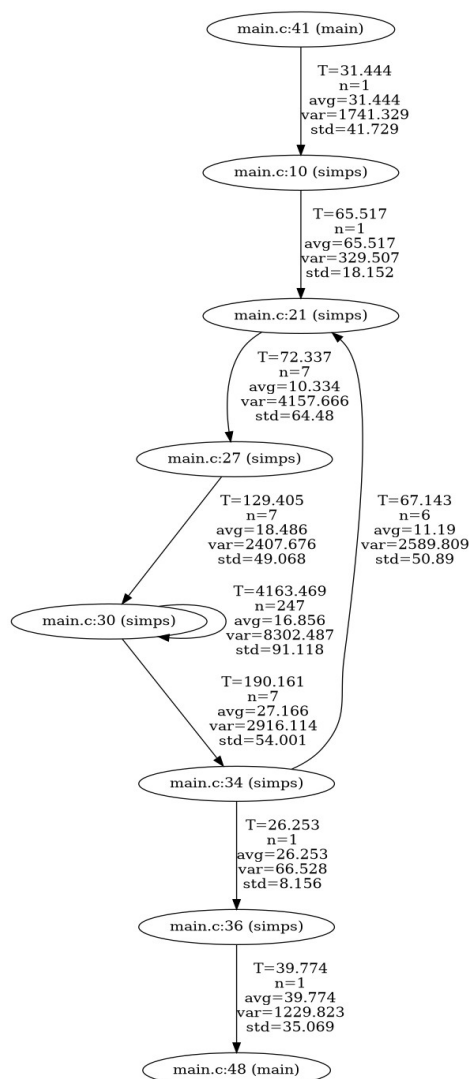


Рисунок 2 - Результаты профилирования программы из лаб. раб. 3 в виде графа

Расчет вероятностей и затрат ресурсов для дуг управляющего графа представлен в табл. 2.

Таблица 2 - Расчет вероятностей и затрат ресурсов для дуг управляющего графа

Дуга	Среднее время исполнения	Вероятность исполнения
1-2	31.44	1
2-3	65.51	1
3-4	10.34	1
4-4	$(129.4+4163.47)/(7+247)=16.9$	$(254-7)/254=0.972$
4-5	27.17	$7/254 = 0.028$
5-3	11.19	$6/(1+6)=0.857$
5-6	26.25	$1/(1+6)=0.143$
6-7	39.77	1

Вероятности для дуг 5-3 и 5-6 были получены экспериментальным путем — для получения необходимой точности необходимо деление области интегрирования на  $2^8$  частей, следовательно нужно переменную *pieces* с начальным значением 2 увеличить в 2 раза 7 раз (7 повторений тела цикла), т. е. дуга 5-6 (выход из цикла *do while*) должна исполниться 1 раз и дуга 5-3 (повторение цикла *do while*) должна исполниться 6 раз.

Вероятности для дуг 4-4 и 4-5 были получены с помощью таблицы 3.

Таблица 3 - Расчет количества повторов внутреннего цикла *for*

Итерация цикла <i>do while</i>	Итераций внутреннего цикла <i>for</i>
1	2
2	4
3	8
4	16
5	32
6	64
7	128

Итого 254 итерации внутреннего цикла *for*. Дуга 4-5 (выход из внутреннего цикла *for*) исполнится 7 раз, т. к. 7 раз исполнится тело цикла *do while*, дуга дуга 4-4 (повторение внутреннего цикла *for*) исполнится  $254-7$  раз.

На рис. 3 представлен граф управления программы с вероятностями перехода и временем исполнения линейных участков (в таком порядке в квадратных скобках).

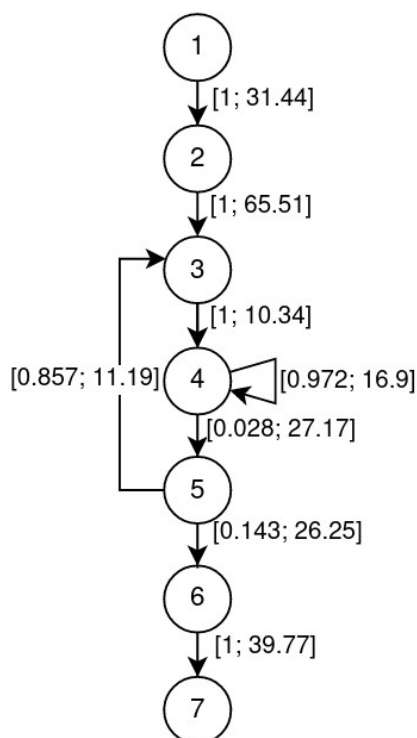


Рисунок 3 - Граф управления программы с нагруженными дугами

3. Было выполнено описание построенной ОГМП в CSA III в виде марковской цепи — см. рис. 4, xml файл см. в приложении Б.

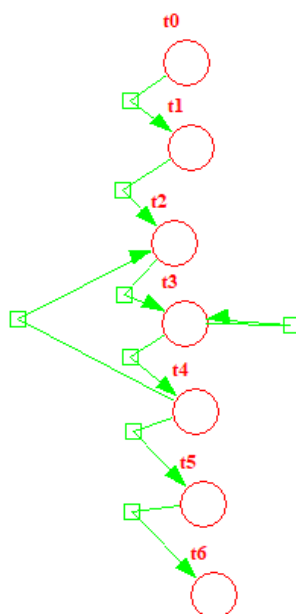
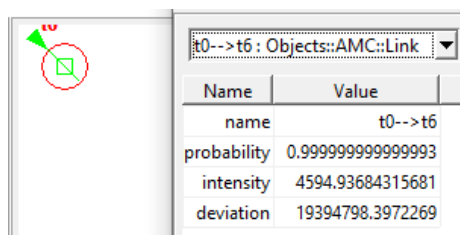


Рисунок 4 - ОГМП в CSA III

После выполнения эквивалентных преобразований пакетом CSA III был получен результат — см. рис. 5.



Name	Value
name	t0-->t6
probability	0.999999999999993
intensity	4594.93684315681
deviation	19394798.3972269

Рисунок 5 — Результат выполнения преобразований пакетом CSA III

Суммированием значений «Общее время» из профиля SAMPLER было получено значение 4785.503, что всего на 4.15% больше значения intensity из результата CSA III.

### **Выводы.**

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 программы. Результаты сравнения этих характеристик с полученными в работе 3 согласуются.

## ПРИЛОЖЕНИЕ А. ПРОГРАММА ИЗ ЛАБ. РАБОТЫ 3

```
1 #include <stdio.h>
2 #include <math.h>
3 #include "sampler.h"
4
5 double fx(double x) {
6     return 1.0 / x;
7 }
8
9 void simps(double lower, double upper, double tol, double *sum) {
10     SAMPLE; // 2
11     int pieces = 2;
12     double
13         delta_x = (upper-lower)/pieces,
14         odd_sum = fx(lower+delta_x),
15         even_sum = 0.0,
16         end_sum = fx(lower) + fx(upper);
17     *sum = (end_sum + 4.0 * odd_sum)*delta_x/3.0;
18     // printf("%5d %f\n", pieces, *sum);
19     double sum1;
20     do {
21         SAMPLE; // 3
22         pieces *= 2;
23         sum1 = *sum;
24         delta_x = (upper-lower)/pieces;
25         even_sum = even_sum + odd_sum;
26         odd_sum = 0.0;
27         SAMPLE; // 4
28         for (int i = 1; i < pieces; i+=2) { // 4 (condition)
29             odd_sum += fx(lower+delta_x*i);
```



```

30     SAMPLE; // 4
31 }
32 *sum = (end_sum + 4.0*odd_sum + 2.0*even_sum)*delta_x/3.0;
33 // printf("%5d %f\n", pieces, *sum);
34     SAMPLE; // 5
35 } while (*sum != sum1 && fabs(*sum-sum1) >= fabs(tol**sum)); // 5
(condition)
36     SAMPLE; // 6
37 }
38
39 int main(int argc, char **argv) {
40     sampler_init(&argc, argv);
41     SAMPLE; // 1
42     const double tol = 1.0e-6;
43     double
44         lower = 1.0,
45         upper = 9.0,
46         sum;
47     simps(lower, upper, tol, &sum);
48     SAMPLE; // 7
49     // printf("\narea=%f\n", sum);
50 }

```

## ПРИЛОЖЕНИЕ Б. XML ФАЙЛ ИЗ CSA III

```
<model type = "Objects::AMC::Model" name = "simps">
  <node type = "Objects::AMC::Top" name = "t0"></node>
  <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>
  <link type = "Objects::AMC::Link" name = "t0-->t1" probability = "1.0"
intensity = "31.44" deviation = "0.0" source = "t0" dest = "t1"></link>
  <link type = "Objects::AMC::Link" name = "t1-->t2" probability = "1.0"
intensity = "65.51" deviation = "0.0" source = "t1" dest = "t2"></link>
  <link type = "Objects::AMC::Link" name = "t2-->t3" probability = "1.0"
intensity = "10.34" deviation = "0.0" source = "t2" dest = "t3"></link>
  <link type = "Objects::AMC::Link" name = "t3-->t4" probability = "0.028"
intensity = "27.17" deviation = "0.0" source = "t3" dest = "t4"></link>
  <link type = "Objects::AMC::Link" name = "t4-->t5" probability = "0.143"
intensity = "26.25" deviation = "0.0" source = "t4" dest = "t5"></link>
  <link type = "Objects::AMC::Link" name = "t5-->t6" probability = "1.0"
intensity = "39.77" deviation = "0.0" source = "t5" dest = "t6"></link>
  <link type = "Objects::AMC::Link" name = "t4-->t2" probability = "0.857"
intensity = "11.19" deviation = "0.0" source = "t4" dest = "t2"></link>
  <link type = "Objects::AMC::Link" name = "t3-->t3" probability = "0.972"
intensity = "16.9" deviation = "0.0" source = "t3" dest = "t3"></link>
</model>
```