

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

ОТЧЕТ
по лабораторной работе №3
по дисциплине «Качество и метрология программного обеспечения»
Тема: Измерение характеристик динамической сложности программ с
помощью профилировщика SAMPLER

Студентка гр. 7304

Юруть Е.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2021

Формулировка задания

1. Ознакомиться с документацией на монитор SAMPLER и выполнить под его управлением тестовые программы `test_cyc.c` и `test_sub.c` с анализом параметров повторения циклов, структуры описания циклов, способов профилирования процедур и проверкой их влияния на точность и чувствительность профилирования.
2. Скомпилировать и выполнить под управлением SAMPLER'a программу на С, разработанную в 1-ой лабораторной работе. Выполнить разбиение программы на функциональные участки и снять профили для двух режимов:
 - а. измерение только полного времени выполнения программы;
 - б. измерение времен выполнения функциональных участков (ФУ).Убедиться, что сумма времен выполнения ФУ соответствует полному времени выполнения программы.
Замечание: следует внимательно подойти к выбору ФУ для получения хороших результатов профилирования.
3. Выявить "узкие места", связанные с ухудшением производительности программы, ввести в программу усовершенствования и получить новые профили. Объяснить смысл введенных модификаций программ.

Ход работы.

1. Профилирование тестовых программ.

1.1. Выполнение тестовой программы test_сус.с под управлением SAMPLER.

Во всей работе будем использовать версию *SAMPLER old*.

Для профилирования программы «Test_сус.cpp» необходимо произвести расстановку макросов-меток, ограничивающих зоны профилирования программы «SAMPLER». Текст программы:

```
File: TEST_CYC.CPP
01: #include "sampler.h"
02: #define Size 10000
03: int i, tmp, dim[Size];
04:
05:
06: void main()
07: {
08:     SAMPLE;
09:     for(i=0;i<Size/10;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp;
10: };
11:     SAMPLE;
12:     for(i=0;i<Size/5;i++){ tmp=dim[0]; dim[0]=dim[i];
13: dim[i]=tmp; };
14:     SAMPLE;
15:     for(i=0;i<Size/2;i++){ tmp=dim[0]; dim[0]=dim[i];
16: dim[i]=tmp; };
17:     SAMPLE;
18:     for(i=0;i<Size;i++) { tmp=dim[0]; dim[0]=dim[i];
19: dim[i]=tmp; };
20:     SAMPLE;
21:     for(i=0;i<Size/10;i++)
22: { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
23:     SAMPLE;
24:     for(i=0;i<Size/5;i++)
25: { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
26:     SAMPLE;
27:     for(i=0;i<Size/2;i++)
28: { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
29:     SAMPLE;
30:     for(i=0;i<Size;i++)
31: { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
32:     SAMPLE;
33:     for(i=0;i<Size/10;i++)
34: { tmp=dim[0];
35: dim[0]=dim[i];
36: dim[i]=tmp;
37: };
38:     SAMPLE;
39:     for(i=0;i<Size/5;i++)
40: { tmp=dim[0];
41: dim[0]=dim[i];
42: dim[i]=tmp;
43: };
44:     SAMPLE;
45:     for(i=0;i<Size/2;i++)
46: { tmp=dim[0];
47: dim[0]=dim[i];
48: dim[i]=tmp;
49: };
50:     SAMPLE;
51:     for(i=0;i<Size;i++)
52: { tmp=dim[0];
53: dim[0]=dim[i];
54: dim[i]=tmp;
55: };
56: }
```

```

48:         { tmp=dim[0];
49:           dim[0]=dim[i];
50:           dim[i]=tmp;
51:         };
52: SAMPLE;
53: }

```

Файл отчёта:

				Список обработанных файлов.	
NN		Имя обработанного файла			
1.		..\TEST\TEST_CYS.CPP			
Таблица с результатами измерений (используется 13 из 416 записей)					
Исх.Поз.	Прием.Поз.	Общее время(мкс)		Кол-во прох.	Среднее время(мкс)
1 :	8 1 :	10	1.68	1	1.68
1 :	10 1 :	12	3.35	1	3.35
1 :	12 1 :	14	6.70	1	6.70
1 :	14 1 :	16	12.57	1	12.57
1 :	16 1 :	19	1.68	1	1.68
1 :	19 1 :	22	3.35	1	3.35
1 :	22 1 :	25	6.70	1	6.70
1 :	25 1 :	28	12.57	1	12.57
1 :	28 1 :	34	1.68	1	1.68
1 :	34 1 :	40	3.35	1	3.35
1 :	40 1 :	46	7.54	1	7.54
1 :	46 1 :	52	13.41	1	13.41

По результатам профилирования можно сделать следующий вывод: структурная организация кода не влияет на время его выполнения, однако на время выполнения цикла влияет количество его итераций (чем больше итераций – тем больше время выполнения цикла, и наоборот).

1.2. Выполнение тестовой программы test_sub.c под управлением SAMPLER.

Для профилирования программы «Test_sub.cpp» необходимо произвести расстановку макросов-меток, ограничивающих зоны профилирования программы «SAMPLER».

Текст программы:

```

1.  #include "sampler.h"
2.
3.  const unsigned Size = 1000;
4.
5.  void TestLoop(int nTimes)
6.  {
7.      static int TestDim[Size];
8.      int tmp;
9.      int iLoop;
10.
11.     while (nTimes > 0)
12.     {
13.         nTimes --;
14.
15.         iLoop = Size;
16.         while (iLoop > 0)
17.         {
18.             iLoop -- ;
19.             tmp = TestDim[0];
20.             TestDim[0] = TestDim[nTimes];
21.             TestDim[nTimes] = tmp;
22.         }
23.     }
24. } /* TestLoop */
25.
26.
27. void main()
28. {
29.     SAMPLE;
30.     TestLoop(Size / 10); // 100 * 1000 повторений
31.     SAMPLE;
32.     TestLoop(Size / 5);  // 200 * 1000 повторений
33.     SAMPLE;
34.     TestLoop(Size / 2);  // 500 * 1000 повторений
35.     SAMPLE;
36.     TestLoop(Size / 1);  // 1000 * 1000 повторений
37.     SAMPLE;
38. }
```

Файл отчёта:

	Список обработанных файлов.			
NN	Имя обработанного файла			
1.	..\TEST\TEST_SUB.CPP			
Таблица с результатами измерений (используется 5 из 416 записей)				
Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 29	1 : 31	92.19	1	92.19
1 : 31	1 : 33	185.02	1	185.02
1 : 33	1 : 35	459.28	1	459.28
1 : 35	1 : 37	950.40	1	950.40

По результатам профилирования можно сделать следующий вывод: при возрастании значения аргумента процедуры TestLoop() линейно увеличивается и время выполнения процедуры (чем больше аргумент, тем выше время выполнения).

2. Профилирование программы на C, разработанной в 1-ой лабораторной работе.

2.1. Измерение полного времени выполнения программы.

Текст программы после расстановки макросов-меток, ограничивающих зоны профилирования программы «SAMPLER»:

```
File: lab1.cpp
01: #include "sampler.h"
02: #include <stdio.h>
03: #include <math.h>
04:
05: double erf(double x)
06: {
07:     const double sqrtpi = 1.7724538;
08:     const double t2 = 0.666666667;
09:     const double t3 = 0.666666667;
10:     const double t4 = 0.07619048;
11:     const double t5 = 0.01693122;
12:     const double t6 = 3.078403E-3;
13:     const double t7 = 4.736005E-4;
14:     const double t8 = 6.314673E-5;
15:     const double t9 = 7.429027E-6;
16:     const double t10 = 7.820028E-7;
17:     const double t11 = 7.447646E-8;
18:     const double t12 = 6.476214E-9;
19:
20:     double x2, sum;
21:     int i;
22:
23:     x2 = x * x;
24:     sum = t5 + x2 * (t6 + x2 * (t7 + x2 * (t8 + x2 * (t9 + x2 *
(t10 + x2 * (t11 + x2 * t12))))));
25:
26:     return 2.0 * exp(-x2) / sqrtpi * (x * (1 + x2 * (t2 + x2 *
(t3 + x2 * (t4 + x2 * sum)))));
27: }
28:
29: double erfc(double x)
30: {
31:     const double sqrtpi = 1.7724538;
32:     double x2, v, sum;
33:
34:     x2 = x * x;
35:     v = 1.0 / (2.0 * x2);
36:     sum = v / (1 + 8 * v / (1 + 9 * v / (1 + 10 * v / (1 + 11 *
v / (1 + 12 * v)))));
37:     sum = v / (1 + 3 * v / (1 + 4 * v / (1 + 5 * v / (1 + 6 * v
/ (1 + 7 * sum)))));
```

```

38:         return 1.0 / (exp(x2) * x * sqrtpi * (1 + v / (1 + 2 *
sum)))));
39: }
40:
41: int main()
42: {
43:     SAMPLE;
44:     double x, er, ec;
45:     int done;
46:     done = 1;
47:     x = 3.0;
48:     do {
49:         x -= 1;
50:         if (x < 0.0){
51:             done = 0;
52:         }
53:         else if (x == 0.0)
54:         {
55:             er = 0.0;
56:             ec = 1.0;
57:         }
58:         else
59:         {
60:             if (x < 1.5)
61:             {
62:
63:                 er = erf(x);
64:                 ec = 1.0 - er;
65:             }
66:             else
67:             {
68:                 ec = erfc(x);
69:                 er = 1.0 - ec;
70:             }
71:         }
72:     } while (done);
73:     SAMPLE;
74:     return 0;
75: }

```

Файл отчёта:

	Список обработанных файлов.			
NN	Имя обработанного файла			
1.	..\TEST\LAB1.CPP			
Таблица с результатами измерений (используется 2 из 416 записей)				
Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 43	1 : 73	24.30	1	24.30

2.2. Измерение времен выполнения функциональных участков программы.

Текст программы после расстановки макросов-меток, ограничивающих зоны профилирования программы «SAMPLER»:

```
File: lab1_1.cpp
01: #include "sampler.h"
02: #include <stdio.h>
03: #include <math.h>
04:
05: double erf(double x)
06: {
07:     SAMPLE;
08:     const double sqrtpi = 1.7724538;
09:     const double t2 = 0.666666667;
10:     const double t3 = 0.666666667;
11:     const double t4 = 0.07619048;
12:     const double t5 = 0.01693122;
13:     const double t6 = 3.078403E-3;
14:     const double t7 = 4.736005E-4;
15:     const double t8 = 6.314673E-5;
16:     const double t9 = 7.429027E-6;
17:     const double t10 = 7.820028E-7;
18:     const double t11 = 7.447646E-8;
19:     const double t12 = 6.476214E-9;
20:
21:     double x2, sum, rez;
22:     int i;
23:     SAMPLE;
24:     x2 = x * x;
25:     sum = t5 + x2 * (t6 + x2 * (t7 + x2 * (t8 + x2 * (t9 + x2 *
(t10 + x2 * (t11 + x2 * t12))))));
26:     rez = 2.0 * exp(-x2) / sqrtpi * (x * (1 + x2 * (t2 + x2 *
(t3 + x2 * (t4 + x2 * sum)))));
27:     SAMPLE;
28:     return rez;
29: }
30:
31: double erfc(double x)
32: {
33:     const double sqrtpi = 1.7724538;
34:     double rez, x2, v, sum;
35:     SAMPLE;
36:     x2 = x * x;
37:     v = 1.0 / (2.0 * x2);
38:     sum = v / (1 + 8 * v / (1 + 9 * v / (1 + 10 * v / (1 + 11 *
v / (1 + 12 * v)))));
39:     sum = v / (1 + 3 * v / (1 + 4 * v / (1 + 5 * v / (1 + 6 * v
/ (1 + 7 * sum)))));
40:     rez = 1.0 / (exp(x2) * x * sqrtpi * (1 + v / (1 + 2 *
sum)));
41:     SAMPLE;
42:     return rez;
43: }
44:
45: int main()
46: {
47:     double x, er, ec;
48:     int done;
49:     SAMPLE;
50:     done = 1;
```



```

51:     x = 3.0;
52:     SAMPLE;
53:     do {
54:         SAMPLE;
55:         x -= 1;
56:         SAMPLE;
57:         if (x < 0.0){
58:             SAMPLE;
59:             done = 0;
60:             SAMPLE;
61:         }
62:         else if (x == 0.0)
63:         {
64:             SAMPLE;
65:             er = 0.0;
66:             ec = 1.0;
67:             SAMPLE;
68:         }
69:         else
70:         {
71:             SAMPLE;
72:             if (x < 1.5)
73:             {
74:                 SAMPLE;
75:                 er = erf(x);
76:                 ec = 1.0 - er;
77:                 SAMPLE;
78:             }
79:             else
80:             {
81:                 SAMPLE;
82:                 ec = erfc(x);
83:                 er = 1.0 - ec;
84:                 SAMPLE;
85:             }
86:             SAMPLE;
87:         }
88:     } while (done);
89:     SAMPLE;
90:     return 0;
91: }

```

Файл отчёта:

Список обработанных файлов.					
NN	Имя обработанного файла				
1.	..\TEST\LAB1_1.CPP				
Таблица с результатами измерений (используется 20 из 416 записей)					
Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)	
1 :	7	1 : 23	3.35	1	3.35
1 :	23	1 : 27	5.87	1	5.87
1 :	27	1 : 77	1.68	1	1.68
1 :	35	1 : 41	10.06	1	10.06
1 :	41	1 : 84	1.68	1	1.68
1 :	49	1 : 52	0.84	1	0.84
1 :	52	1 : 54	0.00	1	0.00
1 :	54	1 : 56	0.84	4	0.21
1 :	56	1 : 71	1.68	2	0.84
1 :	56	1 : 64	0.00	1	0.00
1 :	56	1 : 58	0.84	1	0.84
1 :	58	1 : 60	0.00	1	0.00
1 :	60	1 : 89	0.00	1	0.00
1 :	64	1 : 67	0.84	1	0.84
1 :	67	1 : 54	0.00	1	0.00
1 :	71	1 : 81	0.84	1	0.84
1 :	71	1 : 74	0.00	1	0.00
1 :	74	1 : 7	0.84	1	0.84
1 :	77	1 : 86	0.00	1	0.00
1 :	81	1 : 35	1.68	1	1.68
1 :	84	1 : 86	0.00	1	0.00
1 :	86	1 : 54	0.84	2	0.42

В данном случае сумма времен выполнения ФУ практически соответствует полному времени выполнения программы.

3. Выявление «узких мест» программы, связанных с ухудшением ее производительности.

Проанализировав результаты измерения времени функциональных участков программы, описанных в п.2.2., были сделаны выводы о том, что главными «узкими местами» программы, снижающих ее производительность и время работы, являются использование ненужных переменных и неиспользуемых переменных..

В программу могут быть внесены усовершенствования, нацеленные на увеличение ее производительности:

- удалить неиспользуемую переменную из строки 25.
- удалить ненужную переменную “sum” из строк 21,25,34,38 и 39 и соедините уравнение из строки 25 в строку 26, а также уравнение из строки 38 и 39 в строку 40.
- удалить ненужную переменную “x2” в строке 34 из-за того, что она используется один раз в строке 40, а также удалить ее назначение в строке 36.

3.1. Удаление операции вывода на экран промежуточных значений.

3.1.1. Измерение полного времени выполнения программы.

Текст программы после расстановки макросов-меток, ограничивающих зоны профилирования программы «SAMPLER»:

```
File: lab1_2.cpp
01: #include "sampler.h"
02: #include <stdio.h>
03: #include <math.h>
04:
05: double erf(double x)
06: {
07:     const double sqrtpi = 1.7724538;
08:     const double t2 = 0.666666667;
09:     const double t3 = 0.666666667;
10:     const double t4 = 0.07619048;
11:     const double t5 = 0.01693122;
12:     const double t6 = 3.078403E-3;
13:     const double t7 = 4.736005E-4;
14:     const double t8 = 6.314673E-5;
15:     const double t9 = 7.429027E-6;
16:     const double t10 = 7.820028E-7;
17:     const double t11 = 7.447646E-8;
18:     const double t12 = 6.476214E-9;
19:
20:     double x2, rez;
21:     int i;
22:     x2 = x * x;
23:     rez = 2.0 * exp(-x2) / sqrtpi * (x * (1 + x2 * (t2 + x2 *
(t3 + x2 * (t4 + x2 * (t5 + x2 * (t6 + x2 * (t7 + x2 * (t8 + x2 * (t9
+ x2 * (t10 + x2 * (t11 + x2 * t12))))))))))));
24:     return rez;
25: }
26:
27: double erfc(double x)
28: {
29:     const double sqrtpi = 1.7724538;
```

```

30:     double rez, v;
31:     v = 1.0 / (2.0 * x*x);
32:     rez = 1.0 / (exp(x*x) * x * sqrtpi * (1 + v / (1 + 2 * v /
(1 + 3 * v / (1 + 4 * v / (1 + 5 * v / (1 + 6 * v / (1 + 7 * v / (1 +
8 * v / (1 + 9 * v / (1 + 10 * v / (1 + 11 * v / (1 + 12 *
v))))))))));
33:     return rez;
34: }
35:
36: int main()
37: {
38:     SAMPLE;
39:     double x, er, ec;
40:     int done;
41:     done = 1;
42:     x = 3.0;
43:     do {
44:         x -= 1;
45:         if (x < 0.0){
46:             done = 0;
47:         }
48:         else if (x == 0.0)
49:         {
50:             er = 0.0;
51:             ec = 1.0;
52:         }
53:         else
54:         {
55:             if (x < 1.5)
56:             {
57:                 er = erf(x);
58:                 ec = 1.0 - er;
59:             }
60:             else
61:             {
62:                 ec = erfc(x);
63:                 er = 1.0 - ec;
64:             }
65:         }
66:     } while (done);
67:     SAMPLE;
68:     return 0;
69: }

```

Файл отчёта:

Список обработанных файлов.				
NN	Имя обработанного файла			
1.	..\TEST\LAB1_2.CPP			
Таблица с результатами измерений (используется 2 из 416 записей)				
Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 38	1 : 67	21.45	1	21.45

Для оптимизированной программы общее время выполнения сократилось с 24.30 мкс. до 21.45 мкс(-11,72%).

3.1.2. Измерение времен выполнения функциональных участков программы.

Текст программы после расстановки макросов-меток, ограничивающих зоны профилирования программы «SAMPLER»:

```
File: lab1_3.cpp
01: #include "sampler.h"
02: #include <stdio.h>
03: #include <math.h>
04:
05: double erf(double x)
06: {
07:     SAMPLE;
08:     const double sqrtpi = 1.7724538;
09:     const double t2 = 0.666666667;
10:     const double t3 = 0.666666667;
11:     const double t4 = 0.07619048;
12:     const double t5 = 0.01693122;
13:     const double t6 = 3.078403E-3;
14:     const double t7 = 4.736005E-4;
15:     const double t8 = 6.314673E-5;
16:     const double t9 = 7.429027E-6;
17:     const double t10 = 7.820028E-7;
18:     const double t11 = 7.447646E-8;
19:     const double t12 = 6.476214E-9;
20:
21:     double x2, rez;
22:     int i;
23:     SAMPLE;
24:     x2 = x * x;
25:     rez = 2.0 * exp(-x2) / sqrtpi * (x * (1 + x2 * (t2 + x2 *
(t3 + x2 * (t4 + x2 * (t5 + x2 * (t6 + x2 * (t7 + x2 * (t8 + x2 * (t9
+ x2 * (t10 + x2 * (t11 + x2 * t12))))))))))));
26:     SAMPLE;
27:     return rez;
28: }
29:
30: double erfc(double x)
31: {
32:     const double sqrtpi = 1.7724538;
33:     double rez, v;
34:     SAMPLE;
35:     v = 1.0 / (2.0 * x*x);
36:     rez = 1.0 / (exp(x*x) * x * sqrtpi * (1 + v / (1 + 2 * v /
(1 + 3 * v / (1 + 4 * v / (1 + 5 * v / (1 + 6 * v / (1 + 7 * v / (1 +
8 * v / (1 + 9 * v / (1 + 10 * v / (1 + 11 * v / (1 + 12 *
v))))))))));
37:     SAMPLE;
38:     return rez;
39: }
40:
41: int main()
```

```

42: {
43:     double x, er, ec;
44:     int done;
45:     SAMPLE;
46:     done = 1;
47:     x = 3.0;
48:     SAMPLE;
49:     do {
50:         SAMPLE;
51:         x -= 1;
52:         SAMPLE;
53:         if (x < 0.0){
54:             SAMPLE;
55:             done = 0;
56:             SAMPLE;
57:         }
58:         else if (x == 0.0)
59:         {
60:             SAMPLE;
61:             er = 0.0;
62:             ec = 1.0;
63:             SAMPLE;
64:         }
65:         else
66:         {
67:             SAMPLE;
68:             if (x < 1.5)
69:             {
70:                 SAMPLE;
71:                 er = erf(x);
72:                 ec = 1.0 - er;
73:                 SAMPLE;
74:             }
75:             else
76:             {
77:                 SAMPLE;
78:                 ec = erfc(x);
79:                 er = 1.0 - ec;
80:                 SAMPLE;
81:             }
82:             SAMPLE;
83:         }
84:     } while (done);
85:     SAMPLE;
86:     return 0;
87: }

```

Файл отчёта:

NN		Имя обработанного файла				
1. ..\TEST\LAB1_3.CPP						
Таблица с результатами измерений (используется 20 из 416 записей)						
Исх.Поз.	Прием.Поз.	Общее время(мкс)		Кол-во прох.	Среднее время(мкс)	
1 :	7	1 :	23	3.35	1	3.35
1 :	23	1 :	26	5.03	1	5.03
1 :	26	1 :	73	1.68	1	1.68
1 :	34	1 :	37	9.22	1	9.22
1 :	37	1 :	80	0.84	1	0.84
1 :	45	1 :	48	0.00	1	0.00
1 :	48	1 :	50	0.00	1	0.00
1 :	50	1 :	52	0.84	4	0.21
1 :	52	1 :	67	1.68	2	0.84
1 :	52	1 :	60	0.00	1	0.00
1 :	52	1 :	54	0.00	1	0.00
1 :	54	1 :	56	0.00	1	0.00
1 :	56	1 :	85	0.84	1	0.84
1 :	60	1 :	63	0.84	1	0.84
1 :	63	1 :	50	0.00	1	0.00
1 :	67	1 :	77	0.84	1	0.84
1 :	67	1 :	70	0.84	1	0.84
1 :	70	1 :	7	0.84	1	0.84
1 :	73	1 :	82	0.84	1	0.84
1 :	77	1 :	34	0.84	1	0.84
1 :	80	1 :	82	0.84	1	0.84
1 :	82	1 :	50	0.00	2	0.00

Вывод:

В ходе выполнения данной лабораторной работы было произведено вычисление профиля программы на С с помощью профилировщика Sampler. Произведена оптимизация программы из ЛР1.