

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

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

Студент гр. 6304

\_\_\_\_\_

Тимофеев А.А.

Преподаватель

\_\_\_\_\_

Кириянчиков В.А.

Санкт-Петербург

2020

### **Задание.**

1. Ознакомиться с документацией на 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 C, разработанную в 1-ой лабораторной работе.

Выполнить разбиение программы на функциональные участки и снять профили для двух режимов:

1 - измерение только полного времени выполнения программы;

2 - измерение времен выполнения функциональных участков.

3. Выявить "узкие места", ввести в программы усовершенствования и получить новые профили. Объяснить смысл введенных модификаций программ.

### **Ход работы.**

Использовался старый SAMPLER. Программы компилировались с помощью Borland C++. Компилирование выполнялось на Windows XP, профилирование – в DOSBox.

### **Тестовые программы**

Код программы test\_сус.с с нумерацией строк представлен в приложении А.

### **Результаты профилирования**

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 20	1 : 23	3782.33	1	3782.33
1 : 23	1 : 26	9453.73	1	9453.73
1 : 26	1 : 29	18904.1	1	18904.1
1 : 29	1 : 31	0.84	1	0.84
1 : 31	1 : 37	1892.42	1	1892.42
1 : 37	1 : 43	3780.65	1	3780.65
1 : 43	1 : 49	9452.05	1	9452.05
1 : 49	1 : 55	18905.78	1	18905.78

Была выявлена линейная зависимость результата от количества итераций цикла. Также видно, что времена сильно завышены из-за накладных затрат эмулятора.

Код программы test\_sub.с с нумерацией строк представлен в приложении Б.

### **Результаты профилирования**

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 23	1 : 25	6998.11	1	6998.11
1 : 25	1 : 27	13996.21	1	13996.21
1 : 27	1 : 29	34990.53	1	34990.53
1 : 29	1 : 31	69858.70	1	69858.70

Так как в процедуре выполняется достаточно большое количество итераций, можно проследить зависимость времени выполнения процедуры от количества итераций внутри процедуры. Линейная зависимость времени от количества итераций также прослеживается.

### **Программа из первой лабораторной работы**

Код программы из первой лабораторной работы с нумерацией строк представлен в приложениях В и Г (для измерения полного времени и времени выполнения ФУ соответственно).

### **Результаты профилирования**

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 10	1 : 16	144.15	1	144.15
1 : 16	1 : 26	15554.23	1	15554.23
1 : 26	1 : 36	979.73	1	979.73
1 : 36	1 : 40	7428.05	1	7428.05

Было произведено профилирование только вызова процедуры

### **Результаты профилирования**

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 48	1 : 50	24111.20	1	24111.20

Сумма времён функциональных участков равна 24106.15 и почти равна результату профилирования только вызова процедуры. Выполнение процедуры занимает 24111мкс (0.024с).

Можно сделать оптимизацию – разыменовывание выполняется каждую итерацию цикла, хотя это можно сделать перед циклом. Применим оптимизацию.

Код оптимизированной программы из первой лабораторной работы с нумерацией строк представлен в приложениях Д и Е (для измерения полного времени и времени выполнения ФУ соответственно).

### Результаты профилирования

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 10	1 : 16	142.48	1	142.48
1 : 16	1 : 26	15556.75	1	15556.75
1 : 26	1 : 36	978.06	1	978.06
1 : 36	1 : 41	6313.38	1	6313.38

Было произведено профилирование только вызова процедуры

### Результаты профилирования

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 49	1 : 51	22997.37	1	22997.37

Сумма времён функциональных участков равна 22990.67 и почти равна результату профилирования только вызова процедуры.

Выполнение процедуры занимает 22997мкс (0.023с). Это на  $(24111 - 22997 = 1114\text{мкс})$  быстрее, чем до оптимизации. Сегменты 10-16, 16-26, 26-36 выполняются за такое же время, однако время сегмента 36-41 значительно снизилось на  $(7428 - 6313 = 1115\text{мкс})$ . Это время тратилось на операцию разыменовывания на каждой итерации цикла вместо того, чтобы обращаться напрямую к данным. Оптимизация прошла успешно.

**Вывод:**

В ходе выполнения лабораторной работы был освоен профайлер Sampler. С его помощью была произведена профилировка четырёх программ, написанных на языке С. Производилось профилирование как для функциональных участков процедуры, так и только вызова процедуры. Удалось частично усовершенствовать производительность программы из 1-ой лабораторной работы за счёт выноса разыменовывания из цикла.

## ПРИЛОЖЕНИЕ А

```
1  #include <Sampler.h>
2  #define Size 10000
3  int i, tmp, dim[Size];
4
5  void main()
6  {
7      SAMPLE ;
8      for(i=0;i<Size/10;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
9      SAMPLE ;
10     for(i=0;i<Size/5;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
11     SAMPLE ;
12     for(i=0;i<Size/2;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
13     SAMPLE ;
14     for(i=0;i<Size;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
15     SAMPLE ;
16
17     SAMPLE ;
18     for(i=0;i<Size/10;i++)
19     { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
20     SAMPLE ;
21     for(i=0;i<Size/5;i++)
22     { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
23     SAMPLE ;
24     for(i=0;i<Size/2;i++)
25     { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
26     SAMPLE ;
27     for(i=0;i<Size;i++)
28     { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
29     SAMPLE ;
30
31     SAMPLE ;
32     for(i=0;i<Size/10;i++)
33     { tmp=dim[0];
34       dim[0]=dim[i];
35       dim[i]=tmp;
36     };
37     SAMPLE ;
38     for(i=0;i<Size/5;i++)
39     { tmp=dim[0];
40       dim[0]=dim[i];
41       dim[i]=tmp;
42     };
43     SAMPLE ;
44
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     SAMPLE ;
57 }
58
```

## ПРИЛОЖЕНИЕ Б

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



## ПРИЛОЖЕНИЕ В

```
1  #include <math.h>
2  #include <Sampler.h>
3
4  void linfit2(float *x, float *y, float *y_calc,
5             float *a, float *b, int n){
6      int i;
7      float sum_x, sum_y, sum_xy, sum_x2, sum_y2;
8      float xi, yi, sxy, syy, sxx;
9      float correl_coef, see, sigma_b, sigma_a;
10     sum_x = 0.0;
11     sum_y = 0.0;
12     sum_xy = 0.0;
13     sum_x2 = 0.0;
14     sum_y2 = 0.0;
15     for(i = 0; i < n; i++){
16         xi = x[i];
17         yi = y[i];
18         sum_x = sum_x + xi;
19         sum_y = sum_y + yi;
20         sum_xy = sum_xy + xi * yi;
21         sum_x2 = sum_x2 + xi * xi;
22         sum_y2 = sum_y2 + yi * yi;
23     }
24     sxx = sum_x2 - sum_x * sum_x / n;
25     sxy = sum_xy - sum_x * sum_y / n;
26     syy = sum_y2 - sum_y * sum_y / n;
27     *b = sxy / sxx;
28     *a = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
29     correl_coef = sxy / sqrt(sxx * syy);
30     see = sqrt((sum_y2 - (*a) * sum_y - (*b) * sum_xy) / (n - 2));
31     sigma_b = see / sqrt(sxx);
32     sigma_a = sigma_b * sqrt(sum_x2 / n);
33     for(i = 0; i < n; i++){
34         y_calc[i] = (*a) + (*b) * x[i];
35     }
36 }
37
38 int main(){
39     float x[1000];
40     float y[1000];
41     float y_calc[1000];
42     float a, b;
43     int i;
44     for(i = 0; i < 1000; i++){
45         x[i] = i / 3.0;
46         y[i] = i * i / 3.0;
47     }
48     SAMPLE;
49     linfit2(x, y, y_calc, &a, &b, 1000);
50     SAMPLE;
51 }
52
```

## ПРИЛОЖЕНИЕ Г

```
1  #include <math.h>
2  #include <Sampler.h>
3
4  void linfit2(float *x, float *y, float *y_calc,
5             float *a, float *b, int n){
6      int i;
7      float sum_x, sum_y, sum_xy, sum_x2, sum_y2;
8      float xi, yi, sxy, syy, sxx;
9      float correl_coef, see, sigma_b, sigma_a;
10     SAMPLE;
11     sum_x = 0.0;
12     sum_y = 0.0;
13     sum_xy = 0.0;
14     sum_x2 = 0.0;
15     sum_y2 = 0.0;
16     SAMPLE;
17     for(i = 0; i < n; i++){
18         xi = x[i];
19         yi = y[i];
20         sum_x = sum_x + xi;
21         sum_y = sum_y + yi;
22         sum_xy = sum_xy + xi * yi;
23         sum_x2 = sum_x2 + xi * xi;
24         sum_y2 = sum_y2 + yi * yi;
25     }
26     SAMPLE;
27     sxx = sum_x2 - sum_x * sum_x / n;
28     sxy = sum_xy - sum_x * sum_y / n;
29     syy = sum_y2 - sum_y * sum_y / n;
30     *b = sxy / sxx;
31     *a = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
32     correl_coef = sxy / sqrt(sxx * syy);
33     see = sqrt((sum_y2 - (*a) * sum_y - (*b) * sum_xy) / (n - 2));
34     sigma_b = see / sqrt(sxx);
35     sigma_a = sigma_b * sqrt(sum_x2 / n);
36     SAMPLE;
37     for(i = 0; i < n; i++){
38         y_calc[i] = (*a) + (*b) * x[i];
39     }
40     SAMPLE;
41 }
42
43 int main(){
44     float x[1000];
45     float y[1000];
46     float y_calc[1000];
47     float a, b;
48     int i;
49     for(i = 0; i < 1000; i++){
50         x[i] = i / 3.0;
51         y[i] = i * i / 3.0;
52     }
53     linfit2(x, y, y_calc, &a, &b, 1000);
54 }
55
```

## ПРИЛОЖЕНИЕ Д

```
1  #include <math.h>
2  #include <Sampler.h>
3
4  void linfit2(float *x, float *y, float *y_calc,
5             float *a, float *b, int n){
6      int i;
7      float sum_x, sum_y, sum_xy, sum_x2, sum_y2;
8      float xi, yi, sxy, syy, sxx;
9      float correl_coef, see, sigma_b, sigma_a;
10     sum_x = 0.0;
11     sum_y = 0.0;
12     sum_xy = 0.0;
13     sum_x2 = 0.0;
14     sum_y2 = 0.0;
15     for(i = 0; i < n; i++){
16         xi = x[i];
17         yi = y[i];
18         sum_x = sum_x + xi;
19         sum_y = sum_y + yi;
20         sum_xy = sum_xy + xi * yi;
21         sum_x2 = sum_x2 + xi * xi;
22         sum_y2 = sum_y2 + yi * yi;
23     }
24     sxx = sum_x2 - sum_x * sum_x / n;
25     sxy = sum_xy - sum_x * sum_y / n;
26     syy = sum_y2 - sum_y * sum_y / n;
27     *b = sxy / sxx;
28     *a = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
29     correl_coef = sxy / sqrt(sxx * syy);
30     see = sqrt((sum_y2 - (*a) * sum_y - (*b) * sum_xy) / (n - 2));
31     sigma_b = see / sqrt(sxx);
32     sigma_a = sigma_b * sqrt(sum_x2 / n);
33     int c = *a, d = *b;
34     for(i = 0; i < n; i++){
35         y_calc[i] = c + d * x[i];
36     }
37 }
38
39 int main(){
40     float x[1000];
41     float y[1000];
42     float y_calc[1000];
43     float a, b;
44     int i;
45     for(i = 0; i < 1000; i++){
46         x[i] = i / 3.0;
47         y[i] = i * i / 3.0;
48     }
49     SAMPLE;
50     linfit2(x, y, y_calc, &a, &b, 1000);
51     SAMPLE;
52 }
53
```

## ПРИЛОЖЕНИЕ Е

```
1  #include <math.h>
2  #include <Sampler.h>
3
4  void linfit2(float *x, float *y, float *y_calc,
5             float *a, float *b, int n){
6      int i;
7      float sum_x, sum_y, sum_xy, sum_x2, sum_y2;
8      float xi, yi, sxy, syy, sxx;
9      float correl_coef, see, sigma_b, sigma_a;
10     SAMPLE;
11     sum_x = 0.0;
12     sum_y = 0.0;
13     sum_xy = 0.0;
14     sum_x2 = 0.0;
15     sum_y2 = 0.0;
16     SAMPLE;
17     for(i = 0; i < n; i++){
18         xi = x[i];
19         yi = y[i];
20         sum_x = sum_x + xi;
21         sum_y = sum_y + yi;
22         sum_xy = sum_xy + xi * yi;
23         sum_x2 = sum_x2 + xi * xi;
24         sum_y2 = sum_y2 + yi * yi;
25     }
26     SAMPLE;
27     sxx = sum_x2 - sum_x * sum_x / n;
28     sxy = sum_xy - sum_x * sum_y / n;
29     syy = sum_y2 - sum_y * sum_y / n;
30     *b = sxy / sxx;
31     *a = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
32     correl_coef = sxy / sqrt(sxx * syy);
33     see = sqrt((sum_y2 - (*a) * sum_y - (*b) * sum_xy) / (n - 2));
34     sigma_b = see / sqrt(sxx);
35     sigma_a = sigma_b * sqrt(sum_x2 / n);
36     SAMPLE;
37     int c = *a, d = *b;
38     for(i = 0; i < n; i++){
39         y_calc[i] = c + d * x[i];
40     }
41     SAMPLE;
42 }
43
44 int main(){
45     float x[1000];
46     float y[1000];
47     float y_calc[1000];
48     float a, b;
49     int i;
50     for(i = 0; i < 1000; i++){
51         x[i] = i / 3.0;
52         y[i] = i * i / 3.0;
53     }
54     linfit2(x, y, y_calc, &a, &b, 1000);
55 }
56
```