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 Григорьев И.С.

Преподаватель Кирьянчиков В.А.

Санкт-Петербург

2020

Задание

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:

1 - измерение только полного времени выполнения программы;

2 - измерение времен выполнения функциональных участков (ФУ).

Убедиться, что сумма времен выполнения ФУ соответствует полному времени выполнения программы.

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

Ход работы

Программы транслировались с использованием компилятора Borland C++ v. 3.1. Профилирование выполнялось с помощью sampler_old, который запускался на 32-разрядной виртуальной машине под управлением ОС Windows XP.

Тестовые программы

Код программы test_cyc.c с нумерацией строк представлен в приложении А.

Результаты профилирования программы test_cyc.c:

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	5.03	1	5.03

```
-----------------------------------------------------------------------
1 :  12  1 :  14              9.22            1              9.22
-----------------------------------------------------------------------
1 :  14  1 :  16             17.60            1             17.60
-----------------------------------------------------------------------
1 :  16  1 :  19              1.68            1              1.68
-----------------------------------------------------------------------
1 :  19  1 :  22              4.19            1              4.19
-----------------------------------------------------------------------
1 :  22  1 :  25              8.38            1              8.38
-----------------------------------------------------------------------
1 :  25  1 :  28             17.60            1             17.60
-----------------------------------------------------------------------
1 :  28  1 :  34              1.68            1              1.68
-----------------------------------------------------------------------
1 :  34  1 :  40              4.19            1              4.19
-----------------------------------------------------------------------
1 :  40  1 :  46              8.38            1              8.38
-----------------------------------------------------------------------
1 :  46  1 :  52             18.44            1             18.44
```

Исходя из результата, на время выполнения влияет только количество итераций цикла (линейная зависимость). Циклы с одинаковым количеством итераций демонстрируют почти одно и то же время выполнения.

Код программы test_sub.c с нумерацией строк представлен в приложении Б.

Результаты профилирования программы test_sub.c:

```
  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            145.83            1            145.83
-----------------------------------------------------------------------
1 :  31  1 :  33            295.01            1            295.01
-----------------------------------------------------------------------
1 :  33  1 :  35            739.20            1            739.20
-----------------------------------------------------------------------
1 :  35  1 :  37           1496.00            1           1496.00
-----------------------------------------------------------------------
```

Результаты демонстрируют линейную зависимость времени выполнения функции от количества итераций цикла внутри функции. Время выполнения на два порядка больше, чем в предыдущей программе, не смотря на меньшее количество итераций, – это связано с тем, что в основном цикле каждую итерацию выполняется вложенный цикл.

Программа из первой лабораторной работы

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

Результаты профилирования с измерением полного времени:

Таблица с результатами измерений (используется 2 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 36	1 : 38	59.5	1	59.5

Результаты профилирования с измерением времен ФУ:

Таблица с результатами измерений (используется 8 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 9	1 : 15	3.35	1	3.35
1 : 15	1 : 21	2.51	1	2.51
1 : 21	1 : 24	10.06	12	0.84
1 : 24	1 : 26	3659.13	4095	0.89
1 : 26	1 : 28	9.22	12	0.77
1 : 26	1 : 24	4133.49	4083	1.01
1 : 28	1 : 30	8.38	12	0.70
1 : 30	1 : 21	10.06	11	0.91
1 : 30	1 : 33	0.84	1	0.84

Суммарное время всех ФУ сильно отличается от полного времени выполнения функции только из-за проблемного участка между строками 28 и 31. Время на этом участке большое из-за того, что sampler некорректно считает время для коротких фрагментов программы: время одной итерации он определил как 0.29 мкс, что скорее всего на порядок больше реального значения.

Измененная программа из первой лабораторной работы

В программу были добавлены улучшения: убраны вызовы функции fx и лишние переменные. Измененный код программы из первой лабораторной работы с нумерацией строк представлен в приложениях Д (для измерения полного времени) и Е (для измерения времени выполнения ФУ).

Результаты профилирования с измерением полного времени:

Таблица с результатами измерений (используется 2 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 29	1 : 31	25.14	1	25.14

Общее время выполнения функции уменьшилось в 2 раза, но это верно только для конкретных измерений (на самом деле время выполнения уменьшилось на 10-15%).

Результаты профилирования с измерением времен ФУ:

Таблица с результатами измерений (используется 8 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 13	1 : 19	4.19	1	4.19
1 : 19	1 : 25	1.68	1	1.68
1 : 25	1 : 28	0.84	12	0.07
1 : 28	1 : 31	1190.94	4095	0.29
1 : 31	1 : 33	0.00	12	0.00
1 : 31	1 : 28	1284.80	4083	0.31
1 : 33	1 : 35	0.84	12	0.07
1 : 35	1 : 25	3.35	11	0.30
1 : 35	1 : 38	0.84	1	0.84

Общее время на каждом из участков стало меньше, проблема измерения участка между строками 28-31 также осталась.

Выводы

В ходе лабораторной работы изучен монитор SAMPLER. Выполнено профилирование тестовых программ test_cyc.c и test_sub.c, которое показало линейную зависимость между временем выполнения программы и количеством итераций цикла.

Проанализировано полное время выполнения программы, разработанной в 1-ой лабораторной работе, и время выполнения её ФУ, за счет чего удалось частично усовершенствовать производительность.

ПРИЛОЖЕНИЕ А

TEST_CYC.C

```
1 #include <stdlib.h>
2 #include "Sampler.h"
3 #define Size 10000
4 int i, tmp, dim[Size];
5
6 void main()
7 {
8       SAMPLE;
9       for(i=0;i<Size/10;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
10      SAMPLE;
11      for(i=0;i<Size/5;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
12      SAMPLE;
13      for(i=0;i<Size/2;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
14      SAMPLE;
15      for(i=0;i<Size;i++)  { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
16      SAMPLE;
17      for(i=0;i<Size/10;i++)
18        { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
19      SAMPLE;
20     for(i=0;i<Size/5;i++)
21       { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
22     SAMPLE;
23    for(i=0;i<Size/2;i++)
24       { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
25     SAMPLE;
26     for(i=0;i<Size;i++)
27       { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
28     SAMPLE;
29     for(i=0;i<Size/10;i++)
30       { tmp=dim[0];
31         dim[0]=dim[i];
32         dim[i]=tmp;
33       };
34     SAMPLE;
35     for(i=0;i<Size/5;i++)
36      { tmp=dim[0];
37         dim[0]=dim[i];
38         dim[i]=tmp;
39       };
40     SAMPLE;
41     for(i=0;i<Size/2;i++)
42       { tmp=dim[0];
43         dim[0]=dim[i];
44         dim[i]=tmp;
45       };
46     SAMPLE;
47     for(i=0;i<Size;i++)
48       { tmp=dim[0];
49         dim[0]=dim[i];
50         dim[i]=tmp;
51       };
52     SAMPLE;
53 }
```

ПРИЛОЖЕНИЕ Б

TEST_SUB.C

```
1 #include "Sample.h"
2 const unsigned Size = 1000;
3
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
31   SAMPLE;
32   TestLoop(Size / 5);  // 200 * 1000
33   SAMPLE;
34   TestLoop(Size / 2);  // 500 * 1000
35   SAMPLE;
36   TestLoop(Size / 1);  // 1000* 1000
37   SAMPLE;
38 }
```

ПРИЛОЖЕНИЕ В

Полное время LAB1.CPP

```
1 #include <stdio.h>
2 #include <math.h>
3 #include "sampler.h"
4
5 const double tol = 1.0E-6;
6 double sum, upper, lower;
7
8 double fx(double x) {
9     return 1.0 / x;
10 }
11
12 void trapez(double lower, double upper, double tol) {
13    int pieces = 1;
14    double x, delta_x, end_sum, mid_sum = 0.0, sum1;
15    delta_x = (upper - lower) / pieces;
16    end_sum = fx(lower) + fx(upper);
17    sum = end_sum * delta_x / 2.0;
18    //printf("    1 %.20f\n", sum);
19    do {
20          pieces = pieces * 2;
21          sum1 = sum;
22          delta_x = (upper - lower) / pieces;
23          for (int i = 1; i <= pieces / 2; i++)
24          {
25                x = lower + delta_x * (2.0 * i - 1.0);
26                mid_sum = mid_sum + fx(x);
27          }
28          sum = (end_sum + 2.0 * mid_sum) * delta_x * 0.5;
29          //printf("    %i %.20f\n", pieces, sum);
30    } while (fabs(sum - sum1) > fabs(tol * sum));
31 }
32
33 int main() {
34    lower = 1.0;
35    upper = 9.0;
36    SAMPLE;
37    trapez(lower, upper, tol);
38    SAMPLE;
39    //printf("area = %.20f", sum);
40    return 0;
41 }
```

ПРИЛОЖЕНИЕ Г

Время ФУ LAB1.CPP

```
1. #include <stdio.h>
2. #include <math.h>
3. #include "sampler.h"
4.
5. const double tol = 1.0E-6;
6. double sum, upper, lower;
7.
8. double fx(double x) {
9.     return 1.0 / x;
10.}
11.
12.void trapez(double lower, double upper, double tol) {
13.    SAMPLE;
14.    int pieces = 1;
15.    double x, delta_x, end_sum, mid_sum = 0.0, sum1;
16.    delta_x = (upper - lower) / pieces;
17.    end_sum = fx(lower) + fx(upper);
18.    sum = end_sum * delta_x / 2.0;
19.    SAMPLE;
20.    //printf("    1 %.20f\n", sum);
21.    do {
22.           pieces = pieces * 2;
23.           sum1 = sum;
24.           delta_x = (upper - lower) / pieces;
25.           SAMPLE;
26.           for (int i = 1; i <= pieces / 2; i++)
27.           {
28.                  SAMPLE;
29.                  x = lower + delta_x * (2.0 * i - 1.0);
30.                  mid_sum = mid_sum + fx(x);
31.                  SAMPLE;
32.           }
33.           SAMPLE;
34.           sum = (end_sum + 2.0 * mid_sum) * delta_x * 0.5;
35.           SAMPLE;
36.           //printf("    %i %.20f\n", pieces, sum);
37.    } while (fabs(sum - sum1) > fabs(tol * sum));
38.    SAMPLE;
39. }
40.
41. int main() {
42.    lower = 1.0;
43.    upper = 9.0;
44.    trapez(lower, upper, tol);
45.    //printf("area = %.20f", sum);
46.    return 0;
47. }
```

ПРИЛОЖЕНИЕ Д

Полное время измененной LAB1.C

```
1. #include <stdio.h>
2. #include <math.h>
3. #include "sampler.h"
4.
5. const double tol = 1.0E-6;
6. double sum, upper = 9.0, lower = 1.0;
7.
8. void trapez() {
9.    int pieces = 1;
10.   double delta_x, end_sum, mid_sum = 0.0, sum1;
11.   delta_x = (upper - lower) / pieces;
12.   end_sum = 1.0 / lower + 1.0 / upper;
13.   sum = end_sum * delta_x / 2.0;
14.   //printf("    1 %.20f\n", sum);
15.   do {
16.          pieces *= 2;
17.          sum1 = sum;
18.          delta_x = (upper - lower) / pieces;
19.          for (int i = 1; i <= pieces / 2; i++)
20.          {
21.                mid_sum += 1.0 / (lower + delta_x * (2.0 * i - 1.0));
22.          }
23.          sum = (end_sum + 2.0 * mid_sum) * delta_x * 0.5;
24.          //printf("    %i %.20f\n", pieces, sum);
25.   } while (fabs(sum - sum1) > fabs(tol * sum));
26. }
27.
28. int main() {
29.   SAMPLE;
30.   trapez();
31.   SAMPLE;
32.   //printf("area = %.20f", sum);
33.   return 0;
34. }
```

ПРИЛОЖЕНИЕ Е

Время ФУ измененной LAB1.C

```
1. #include <stdio.h>
2. #include <math.h>
3. #include "sampler.h"
4.
5. const double tol = 1.0E-6;
6. double sum, upper = 9.0, lower = 1.0;
7.
8. void trapez() {
9.    SAMPLE;
10.   int pieces = 1;
11.   double delta_x, end_sum, mid_sum = 0.0, sum1;
12.   delta_x = (upper - lower) / pieces;
13.   end_sum = 1.0 / lower + 1.0 / upper;
14.   sum = end_sum * delta_x / 2.0;
15.   SAMPLE;
16.   //printf("    1 %.20f\n", sum);
17.   do {
18.         pieces *= 2;
19.         sum1 = sum;
20.         delta_x = (upper - lower) / pieces;
21.         SAMPLE;
22.         for (int i = 1; i <= pieces / 2; i++)
23.         {
24.               SAMPLE;
25.               mid_sum += 1.0 / (lower + delta_x * (2.0 * i - 1.0));
26.               SAMPLE;
27.         }
28.         SAMPLE;
29.         sum = (end_sum + 2.0 * mid_sum) * delta_x * 0.5;
30.         SAMPLE;
31.         //printf("    %i %.20f\n", pieces, sum);
32.   } while (fabs(sum - sum1) > fabs(tol * sum));
33.   SAMPLE;
34. }
35.
36. int main() {
37.   trapez();
38.   //printf("area = %.20f", sum);
39.   return 0;
40. }
```