**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 _____ Зубов К.А.

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 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 (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 | 1.72 | 1 | 1.72 |
| 1 : 10 | 1 : 12 | 5.13 | 1 | 5.13 |
| 1 : 12 | 1 : 14 | 9.30 | 1 | 9.30 |
| 1 : 14 | 1 : 16 | 17.40 | 1 | 17.40 |
| 1 : 16 | 1 : 19 | 1.56 | 1 | 1.56 |
| 1 : 19 | 1 : 22 | 4.23 | 1 | 4.23 |
| 1 : 22 | 1 : 25 | 8.35 | 1 | 8.35 |
| 1 : 25 | 1 : 28 | 17.58 | 1 | 17.58 |
| 1 : 28 | 1 : 34 | 1.68 | 1 | 1.68 |
| 1 : 34 | 1 : 40 | 4.25 | 1 | 4.25 |
| 1 : 40 | 1 : 46 | 8.36 | 1 | 8.36 |
| 1 : 46 | 1 : 52 | 18.39 | 1 | 18.39 |

Исходя из результатов, на время выполнения влияет 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             146.15            1             146.15
-----------------------------------------------------------------------
1 :  31  1 :  33             282.31            1             282.31
-----------------------------------------------------------------------
1 :  33  1 :  35             771.53            1             771.53
-----------------------------------------------------------------------
1 :  35  1 :  37            1420.04            1            1420.04
-----------------------------------------------------------------------
```

Результаты показывают линейную зависимость времени выполнения функции от количества итераций цикла внутри этой функции. Из-за того, что в основном цикле каждую итерацию выполняется вложенный цикл - время выполнения на два порядка больше, чем в предыдущей программе.

**Программа из первой лабораторной работы**

Код программы из первой лабораторной работы 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 :  35  1 :  37              25.98            1              25.98
-----------------------------------------------------------------------
```

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 :  24               1.68            1              1.68
-------------------------------------------------------------------
 1 :  24  1 :  27               4.19           11              0.38
-------------------------------------------------------------------
 1 :  27  1 :  30            1250.44         2047              0.61
-------------------------------------------------------------------
 1 :  30  1 :  32               7.54           11              0.69
 1 :  30  1 :  27            1971.20         2036              0.97
-------------------------------------------------------------------
 1 :  32  1 :  34              12.57           11              1.14
-------------------------------------------------------------------
 1 :  34  1 :  24               5.03           10              0.50
 1 :  34  1 :  36               0.84            1              0.84
-------------------------------------------------------------------
```

Суммарное время всех ФУ сильно отличается от полного времени выполнения функции. Это может быть связано с вызовом функции fx. Заменим эти вызовы на содержимое функции.

**Измененная программа из первой лабораторной работы**

В программу были добавлены улучшения: убраны вызовы функции fx и лишние переменные. Измененный код программы 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 на 6.5%.

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 | 4.19 | 1 | 4.19 |
| 1 : 15 | 1 : 20 | 1.68 | 1 | 1.68 |
| 1 : 20 | 1 : 23 | 3.4 | 11 | 0.30 |
| 1 : 23 | 1 : 25 | 1257.14 | 2047 | 0.61 |
| 1 : 25 | 1 : 27 | 7.54 | 11 | 0.69 |
| 1 : 25 | 1 : 23 | 2245.26 | 2036 | 1.10 |
| 1 : 27 | 1 : 29 | 8.38 | 11 | 0.76 |
| 1 : 29 | 1 : 20 | 3.54 | 10 | 0.34 |
| 1 : 29 | 1 : 31 | 0.00 | 1 | 0.00 |

Общее время на нескольких участках стало ниже. Возможно, что sampler некорректно считает время для коротких фрагментов программы, поэтому на некоторых участках среднее время немного выше.

**Выводы**

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

Благодаря анализу полного времени выполнения программы, разработанной в 1-ой лабораторной работе, и времени выполнения её ФУ, 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;  }
```

# 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 double sum = 0.0;
6 double upper, lower;
7 const double tol = 1.0E-6;
8 double fx(double x) {
9     return 1.0 / sqrt(x);
10}
11
12void trap2(double lower, double upper, double tol) {
13    int pieces = 1;
14    double x,delta_x,end_sum,mid_sum,sum1;
15    delta_x = ( upper - lower )/pieces;
16    end_sum = fx(lower) + fx(upper);
17    sum = end_sum * delta_x/2.0;
18    mid_sum = 0.0;
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    } while (fabs(sum - sum1) > fabs(tol * sum));
30}
31
32int main() {
33    lower = 1.0;
34    upper = 9.0;
35    SAMPLE;
36    trap2(lower, upper, tol);
37    SAMPLE;
38    return 0;
```

```
39}
```

# ПРИЛОЖЕНИЕ Г

## Время ФУ LAB1.CPP

```
1 #include <stdio.h>
2 #include <math.h>
3 #include "Sampler.h"
4
5 double sum = 0.0;
6 double upper, lower;
7 const double tol = 1.0E-6;
8 double fx(double x) {
9     return 1.0 / sqrt(x);
10}
11
12void trap2(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    do {
21          pieces = pieces * 2;
22          sum1 = sum;
23          delta_x = (upper - lower) / pieces;
24          SAMPLE;
25          for (int i = 1; i <= pieces/2; i++)
26          {
27                SAMPLE;
28                x = lower + delta_x * (2.0 * i - 1.0);
29                mid_sum = mid_sum + fx(x);
30                SAMPLE;
31          }
32          SAMPLE;
```

```
33            sum = ( end_sum + 2.0 * mid_sum ) * delta_x * 0.5;
34            SAMPLE;
35    } while (fabs(sum - sum1) > fabs(tol * sum));
36    SAMPLE;
37}
38
39int main() {
40    lower = 1.0;
41    upper = 9.0;
42    trap2(lower, upper, tol);
43    return 0;
44}
```

# ПРИЛОЖЕНИЕ Д

## Полное время измененной LAB1.C

```
1 #include <stdio.h>
2 #include <math.h>
3 #include "Sampler.h"
4 double sum = 0.0;
5 double upper = 9.0, lower = 1.0;
6 const double tol = 1.0E-6;
7
8 void trap2() {
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/sqrt(lower) + 1.0/sqrt(upper);
13    sum = end_sum * delta_x/2.0;
14
15    do {
16          pieces = 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/sqrt(lower + delta_x * (2.0 * i - 1.0));
22          }
23          sum = ( end_sum + 2.0 * mid_sum ) * delta_x * 0.5;
24
25    } while (fabs(sum - sum1) > fabs(tol * sum));
26}
27
28int main() {
29    SAMPLE;
30    trap2();
31    SAMPLE;
32    return 0;
33}
```

# ПРИЛОЖЕНИЕ Е

## Время ФУ измененной LAB1.C

```
1 #include <stdio.h>
2 #include <math.h>
3 #include "Sampler.h"
4 double sum = 0.0;
5 double upper = 9.0, lower = 1.0;
6 const double tol = 1.0E-6;
7
8 void trap2() {
9     SAMPLE;
10    int pieces = 1;
11    double x,delta_x,end_sum,mid_sum = 0.0,sum1;
12    delta_x = ( upper - lower )/pieces;
13    end_sum = 1.0/sqrt(lower) + 1.0/sqrt(upper);
14    sum = end_sum * delta_x/2.0;
15    SAMPLE;
16    do {
17          pieces = pieces * 2;
18          sum1 = sum;
19          delta_x = (upper - lower) / pieces;
20          SAMPLE;
21          for (int i = 1; i <= pieces/2; i++)
22          {
23                SAMPLE;
24                mid_sum += 1.0/sqrt(lower + delta_x * (2.0 * i - 1.0));
25                SAMPLE;
26          }
27          SAMPLE;
28          sum = ( end_sum + 2.0 * mid_sum ) * delta_x * 0.5;
29          SAMPLE;
30    } while (fabs(sum - sum1) > fabs(tol * sum));
31    SAMPLE;
32}
33
34int main() {
35    trap2();
36    return 0;
37}
```