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 _____ Зыль С.Е.

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.

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.

Среда разработки.

Для разработки, компиляции и профилирования программ использовались следующие программы:

- Borland C++ Version 3.1

Использовалась старая версия программы и запускалась она на виртуальной машине, на которой стоит операционная система Microsoft Windows XP

Анализ тестовых программ.

Код программы TEST_CYC.CPP

```
01 #include <stdlib.h>
02 #include "Sampler.h"
03 #define Size 10000
04
05 int i, tmp, dim[Size];
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   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   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
25   for(i=0;i<Size/2;i++)
26        { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
27   SAMPLE;
28
29   for(i=0;i<Size;i++)
30        { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
31   SAMPLE;
32
33   SAMPLE;
34   for(i=0;i<Size/10;i++)
35   {
36        tmp=dim[0];
37        dim[0]=dim[i];
```

```
38          dim[i]=tmp;
39    };
40
41    SAMPLE;
42
43    for(i=0;i<Size/5;i++)
44    {
45          tmp=dim[0];
46          dim[0]=dim[i];
47          dim[i]=tmp;
48    };
49
50    SAMPLE;
51
52    for(i=0;i<Size/2;i++)
53    {
54          tmp=dim[0];
55          dim[0]=dim[i];
56          dim[i]=tmp;
57    };
58
59    SAMPLE;
60    for(i=0;i<Size;i++)
61    {
62    tmp=dim[0];
63    dim[0]=dim[i];
64    dim[i]=tmp;
65    };
66    SAMPLE;
67 }
```

Результат профилирования TEST_CYC.CPP

```
---------------------------------------------------------------------
  NN          Имя обработанного файла
---------------------------------------------------------------------
  1.    TEST_CYC.CPP
---------------------------------------------------------------------
```

Таблица с результатами измерений (используется 15 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 8	1 : 10	3.55	1	3.55
1 : 10	1 : 12	6.07	1	6.07
1 : 12	1 : 14	17.39	1	17.39
1 : 14	1 : 16	36.43	1	36.43
1 : 16	1 : 17	0.00	1	0.00
1 : 17	1 : 20	3.39	1	3.39
1 : 20	1 : 23	5.93	1	5.93
1 : 23	1 : 27	16.82	1	16.82
1 : 27	1 : 31	35.67	1	35.67
1 : 31	1 : 33	0.00	1	0.00
1 : 33	1 : 41	3.39	1	3.39
1 : 41	1 : 50	6.07	1	6.07
1 : 50	1 : 59	17.22	1	17.22
1 : 59	1 : 66	35.27	1	35.27

По полученным результатам видно, что время циклов, у которых одинаковое количество итераций, примерно одинаковое, а это означает, что от записи цикла не зависит время его выполнения.

Код программы TEST_SUB.CPP

```
01 #include <stdlib.h>
02 #include "Sampler.h"
03
04 const unsigned Size = 1000;
05
06
07 void TestLoop(int nTimes)
08 {
09   static int TestDim[Size];
10   int tmp;
```

```
11   int iLoop;
12
13   while (nTimes > 0)
14   {
15     nTimes --;
16
17     iLoop = Size;
18     while (iLoop > 0)
19     {
20       iLoop -- ;
21       tmp = TestDim[0];
22       TestDim[0] = TestDim[nTimes];
23       TestDim[nTimes] = tmp;
24     }
25   }
26 } /* TestLoop */
27
28
29 void main()
30 {
31 SAMPLE;
32    TestLoop(Size / 10); // 100 * 1000  повторений
33 SAMPLE;
34    TestLoop(Size / 5);  // 200 * 1000  повторений
35 SAMPLE;
36    TestLoop(Size / 2);  // 500 * 1000  повторений
37 SAMPLE;
38    TestLoop(Size / 1);  // 1000* 1000  повторений
39 SAMPLE;
40 }
```

Результат профилирования TEST_SUB.CPP

Список обработанных файлов.

NN	Имя обработанного файла
1.	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 : 31	1 : 33	304.65	1	304.65

1 : 33	1 : 35	624.92	1	624.92
1 : 35	1 : 37	1558.74	1	1558.74
1 : 37	1 : 39	3025.92	1	3025.92

По полученным данным видно, что прослеживается линейная зависимость времени выполнения каждого вызова функции от количества итераций цикла в функции.

Профилирование программы на Си

Текст программы на Си для измерения общего времени.

```
01 #include <cstdlib>
02 #include <ctime>
03 #include "Sampler.h"
04
05 const int MAX = 100;
06
07 void sort1(double * a, int n)
08 {
09   double hold;
10   for (int i = 0; i < n; i++) {
11        for (int j = i + 1; j < n; j++) {
12             if (a[j] > a[i]) {
13                  hold = a[i];
14                  a[i] = a[j];
15                  a[j] = hold;
16             }
17        }
18   }
19 }
20
21
22 void swap(double * a, int p, int q)
23 {
24   double hold = a[p];
25   a[p] = a[q];
26   a[q] = hold;
27 }
28
29 void sort2(double * a, int n)
30 {
31   int noChange = 1;
32   do {
33        noChange = 1;
34        for (int i = 0; i < n - 1; i++) {
35             if (a[i] > a[i + 1]) {
36                  swap(a, i, i + 1);
37                  noChange = 0;
38             }
39        }
40   } while (noChange != 1);
41 }
42
43
44 int main(int argc, char const *argv[])
45 {
46   double * forFirstSort = (double *)calloc(MAX, sizeof(double));
47   double * forSecondSort = (double *)calloc(MAX, sizeof(double));
48   srand((unsigned)time(NULL));
49
```

```
50   double randNuber = 0;
51   for (int i = 0; i < MAX; i++) {
52        int rn = (rand() % 1000 + 1);
53        randNuber = (double) rn;
54        forFirstSort[i] = randNuber;
55        forSecondSort[i] = randNuber;
56   }
57   SAMPLE;
58   sort1(forFirstSort, MAX);
59   SAMPLE;
60   sort2(forSecondSort, MAX);
61   SAMPLE;
62   return 0;
63 }
```

Результат профилирования

Список обработанных файлов.

NN	Имя обработанного файла
1.	KMPO.CPP

Таблица с результатами измерений (используется 3 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 57	1 : 59	73.87	1	73.87
1 : 59	1 : 61	155.08	1	155.08

Текст программы для измерения функциональных участков

```
01 #include <cstdlib>
02 #include <ctime>
03 #include "Sampler.h"
04 const int MAX = 100;
05 void sort1(double * a, int n)
06 {
07   double hold;
08   SAMPLE;
09   for (int i = 0; i < n; i++)
```

```
10   {
11        SAMPLE;
12        for (int j = i + 1; j < n; j++)
13        {
14             SAMPLE;
15             if (a[j] > a[i])
16             {
17                  SAMPLE;
18                  hold = a[i];
19                  a[i] = a[j];
20                  a[j] = hold;
21                  SAMPLE;
22             }
23             SAMPLE;
24        }
25        SAMPLE;
26   }
27   SAMPLE;
28 }
29
30
31 void swap(double * a, int p, int q)
32 {
33   double hold = a[p];
34   a[p] = a[q];
35   a[q] = hold;
36 }
37
38 void sort2(double * a, int n)
39 {
40   int noChange = 1;
41   SAMPLE;
42   do
43   {
44        SAMPLE;
45        noChange = 1;
46        SAMPLE;
47        for (int i = 0; i < n - 1; i++)
48        {
49             SAMPLE;
50             if (a[i] > a[i + 1])
51             {
52                  SAMPLE;
53                  swap(a, i, i + 1);
54                  noChange = 0;
55                  SAMPLE;
56             }
57             SAMPLE;
58        }
59        SAMPLE;
60   } while (noChange != 1);
61   SAMPLE;
62 }
```

```
63
64
65 int main(int argc, char const *argv[])
66 {
67   double * forFirstSort = (double *)calloc(MAX, sizeof(double));
68   double * forSecondSort = (double *)calloc(MAX, sizeof(double));
69   srand((unsigned)time(NULL));
70
71   double randNuber = 0;
72   for (int i = 0; i < MAX; i++) {
73        int rn = (rand() % 1000 + 1);
74        randNuber = (double) rn;
75        forFirstSort[i] = randNuber;
76        forSecondSort[i] = randNuber;
77   }
78   sort1(forFirstSort, MAX);
79   sort2(forSecondSort, MAX);
80   return 0;
81 }
```

Результат профилирования

Список обработанных файлов.

NN	Имя обработанного файла
1.	KMPO_F.CPP

Таблица с результатами измерений (используется 17 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 8	1 : 11	0.00	1	0.00
1 : 11	1 : 14	1.68	99	0.02
1 : 14	1 : 17	1.68	1569	0.00
1 : 14	1 : 23	2.52	3381	0.00
1 : 17	1 : 21	66.69	1569	0.04
1 : 21	1 : 23	0.00	1569	0.00

```
 1 :  23  1 :  14          2.52        4851          0.00
 1 :  23  1 :  25          0.00          99          0.00
-----------------------------------------------------------------
 1 :  25  1 :  11          2.52          98          0.03
 1 :  25  1 :  27          0.00           1          0.00
-----------------------------------------------------------------
 1 :  27  1 :  41          1.68           1          1.68
-----------------------------------------------------------------
 1 :  41  1 :  44          0.00           1          0.00
-----------------------------------------------------------------
 1 :  44  1 :  46          1.68          99          0.02
-----------------------------------------------------------------
 1 :  46  1 :  49          0.00          99          0.00
-----------------------------------------------------------------
 1 :  49  1 :  52          0.00        2494          0.00
 1 :  49  1 :  57          4.20        7307          0.00
-----------------------------------------------------------------
 1 :  52  1 :  55        134.86        2494          0.05
-----------------------------------------------------------------
 1 :  55  1 :  57          0.00        2494          0.00
-----------------------------------------------------------------
 1 :  57  1 :  49         12.62        9702          0.00
 1 :  57  1 :  59          0.00          99          0.00
-----------------------------------------------------------------
 1 :  59  1 :  44          6.35          98          0.06
 1 :  59  1 :  61          0.00           1          0.00
-----------------------------------------------------------------
```

Для первой функции время по функциональным участкам составило 77.61, что больше на 5% общего времени (73.87) выполнения этой функции. Наибольшее количество временных ресурсов тратится на обмен местами двух элементов массива.

Для второй функции время по функциональным участкам составило 159.71, что больше на 3% общего времени (155.08) выполнения этой функции. Наибольшее количество временных ресурсов тратится на вызов функции, в которой происходит обмен местами двух элементов массива. Для улучшения данной функции уберем вспомогательную функцию и сделаем обмен в основной.

Профилирование улучшенной программы на Си

Текст программы на Си для измерения общего времени.

```
01 #include <cstdlib>
02 #include <ctime>
03 #include "Sampler.h"
04 const int MAX = 100;
05 void sort1(double * a, int n)
06 {
07   double hold;
08   for (int i = 0; i < n; i++)
09   {
10        for (int j = i + 1; j < n; j++)
11        {
12             if (a[j] > a[i])
13             {
14                  hold = a[i];
15                  a[i] = a[j];
16                  a[j] = hold;
17             }
18        }
19   }
20 }
21
22 void sort2(double * a, int n)
23 {
24   int noChange = 1;
25   do
26   {
27        noChange = 1;
28        for (int i = 0; i < n - 1; i++)
29        {
30             if (a[i] > a[i + 1])
31             {
32                  double hold = a[i];
33                  a[i] = a[i + 1];
34                  a[i + 1] = hold;
35                  noChange = 0;
36             }
37        }
38   } while (noChange != 1);
39 }
40
41
42 int main(int argc, char const *argv[])
43 {
44   double * forFirstSort = (double *)calloc(MAX, sizeof(double));
45   double * forSecondSort = (double *)calloc(MAX, sizeof(double));
46   srand((unsigned)time(NULL));
47
48   double randNuber = 0;
49   for (int i = 0; i < MAX; i++) {
```

```
50         int rn = (rand() % 1000 + 1);
51         randNuber = (double) rn;
52         forFirstSort[i] = randNuber;
53         forSecondSort[i] = randNuber;
54    }
55    SAMPLE;
56    sort1(forFirstSort, MAX);
57    SAMPLE;
58    sort2(forSecondSort, MAX);
59    SAMPLE;
60    return 0;
61 }
```

Результат профилирования

Список обработанных файлов.

NN	Имя обработанного файла
1.	KMPO.CPP

Таблица с результатами измерений (используется 3 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 55	1 : 57	74.23	1	74.23
1 : 57	1 : 59	122.37	1	122.37

Время выполнения первой функции изменилось не сильно, а время второй функции уменьшилось на 32.71 мкс (примерно на 1/5 общего времени выполнения).

Текст программы на Си для измерения времени функциональных участков.

```
01 #include <cstdlib>
02 #include <ctime>
03 #include "Sampler.h"
04 const int MAX = 100;
05 void sort1(double * a, int n)
06 {
07   double hold;
```

```
08   SAMPLE;
09   for (int i = 0; i < n; i++)
10   {
11        SAMPLE;
12        for (int j = i + 1; j < n; j++)
13        {
14             SAMPLE;
15             if (a[j] > a[i])
16             {
17                  SAMPLE;
18                  hold = a[i];
19                  a[i] = a[j];
20                  a[j] = hold;
21                  SAMPLE;
22             }
23             SAMPLE;
24        }
25        SAMPLE;
26   }
27   SAMPLE;
28 }
29
30 void sort2(double * a, int n)
31 {
32   int noChange = 1;
33   SAMPLE;
34   do
35   {
36        SAMPLE;
37        noChange = 1;
38        SAMPLE;
39        for (int i = 0; i < n - 1; i++)
40        {
41             SAMPLE;
42             if (a[i] > a[i + 1])
43             {
44                  SAMPLE;
45                  double hold = a[i];
46                  a[i] = a[i + 1];
47                  a[i + 1] = hold;
48                  noChange = 0;
49                  SAMPLE;
50             }
51             SAMPLE;
52        }
53        SAMPLE;
54   } while (noChange != 1);
55   SAMPLE;
56 }
57
58
59 int main(int argc, char const *argv[])
60 {
```

```
61    double * forFirstSort = (double *)calloc(MAX, sizeof(double));
62    double * forSecondSort = (double *)calloc(MAX, sizeof(double));
63    srand((unsigned)time(NULL));
64
65    double randNuber = 0;
66    for (int i = 0; i < MAX; i++) {
67         int rn = (rand() % 1000 + 1);
68         randNuber = (double) rn;
69         forFirstSort[i] = randNuber;
70         forSecondSort[i] = randNuber;
71    }
72    sort1(forFirstSort, MAX);
73    sort2(forSecondSort, MAX);
74    return 0;
75 }
```

Результат профилирования

Список обработанных файлов.

NN	Имя обработанного файла
1.	KMPO_F.CPP

Таблица с результатами измерений (используется 17 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 8	1 : 11	0.00	1	0.00
1 : 11	1 : 14	1.80	99	0.02
1 : 14	1 : 17	0.00	1569	0.00
1 : 14	1 : 23	3.00	3381	0.00
1 : 17	1 : 21	64.73	1569	0.04
1 : 21	1 : 23	0.00	1569	0.00
1 : 23	1 : 14	2.41	4851	0.00
1 : 23	1 : 25	0.00	99	0.00

```
 1 :  25  1 :  11          3.95        98          0.04
 1 :  25  1 :  27          0.00         1          0.00
-------------------------------------------------------------------
 1 :  27  1 :  33          1.68         1          1.68
-------------------------------------------------------------------
 1 :  33  1 :  36          0.00         1          0.00
-------------------------------------------------------------------
 1 :  36  1 :  38          1.86        99          0.02
-------------------------------------------------------------------
 1 :  38  1 :  41          0.00        99          0.00
-------------------------------------------------------------------
 1 :  41  1 :  44          0.00      2494          0.00
 1 :  41  1 :  51          3.52      7307          0.00
-------------------------------------------------------------------
 1 :  44  1 :  49        104.37      2494          0.04
-------------------------------------------------------------------
 1 :  49  1 :  51          0.00      2494          0.00
-------------------------------------------------------------------
 1 :  51  1 :  41         12.46      9702          0.00
 1 :  51  1 :  53          0.00        99          0.00
-------------------------------------------------------------------
 1 :  53  1 :  36          3.52        98          0.04
 1 :  53  1 :  55          0.00         1          0.00
-------------------------------------------------------------------
```

В последнем протоколе можно заметить, что время выполнения одного фрагмента обмена местами двух элементов теперь одинаковы.

Время второй функции больше из-за большего количества итераций одного из циклов, а соответственно и количества перестановок.

Выводы.

В данной лабораторной работе мы ознакомились с документацией на монитор SAMPLER и выполнили под его управлением тестовые программы test_cyc.c и test_sub.c. Также выполнили профилирование программы на Си из первой лабораторной работы. После выявления «узких мест» выполнили улучшение программы.