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

Студентка гр. 7304 _____ Каляева А.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

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:

Для выполнения лабораторной работы был выбран монитор Sampler_old. Программы были скомпилированы Borland C++ 3.1 на DosBox, профилирование было проведено на DosBox.

1. Профилирование тестовых файлов

Исходный код файла TEST_CYC.CPP с нумерацией строк представлен в приложении А.

Результаты профилирования:

```
   Отчет о результатах измерений для программы Q.EXE.

   Создан программой Sampler ( версия от Feb 15 1999 )
       1995-98 (c) СПбГЭТУ, Мойсейчук Леонид.


       Список обработанных файлов.

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


   Таблица с результатами измерений ( используется 13 из 416 записей )

-------------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-------------------------------------------------------------------------
 1 :   9  1 :  11            4334.64            1            4334.64
-------------------------------------------------------------------------
 1 :  11  1 :  13            8675.98            1            8675.98
-------------------------------------------------------------------------
 1 :  13  1 :  15           21671.50            1           21671.50
-------------------------------------------------------------------------
 1 :  15  1 :  17           43348.03            1           43348.03
-------------------------------------------------------------------------
 1 :  17  1 :  20            4336.31            1            4336.31
-------------------------------------------------------------------------
 1 :  20  1 :  23            8668.43            1            8668.43
-------------------------------------------------------------------------
 1 :  23  1 :  26           21673.18            1           21673.18
-------------------------------------------------------------------------
 1 :  26  1 :  29           43348.03            1           43348.03
-------------------------------------------------------------------------
 1 :  29  1 :  35            4335.47            1            4335.47
-------------------------------------------------------------------------
 1 :  35  1 :  41            8669.27            1            8669.27
-------------------------------------------------------------------------
 1 :  41  1 :  47           21671.50            1           21671.50
-------------------------------------------------------------------------
 1 :  47  1 :  53           43348.87            1           43348.87
-------------------------------------------------------------------------
```

Исходный код файла TEST_SUB.CPP с нумерацией строк представлен в приложении Б.

Результаты профилирования:

```
    Отчет о результатах измерений для программы Q1.EXE.

    Создан программой Sampler ( версия от Feb 15 1999 )
        1995-98 (c) СПбГЭТУ, Мойсейчук Леонид.



        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 :  24  1 :  26          433698.18             1          433698.18
-------------------------------------------------------------------------
 1 :  26  1 :  28          867393.02             1          867393.02
-------------------------------------------------------------------------
 1 :  28  1 :  30         2168475.00             1         2168475.00
-------------------------------------------------------------------------
 1 :  30  1 :  32         4336944.97             1         4336944.97
-------------------------------------------------------------------------
```

Согласно результатам профилирования можно заметить, что время для циклов с одинаковым количеством итераций примерно одинаково. Исходя из этого можно сделать вывод, что от записи цикла не зависит время его выполнения. Также прослеживается линейная зависимость времени выполнения каждого вызова функции от количества итераций цикла в функции.

2. Профилирование файла из лабораторной работы

Исходный код файла SHELL.CPP с нумерацией строк для измерения общего времени представлен в приложении В.

Результаты профилирования:

Отчет о результатах измерений для программы Q2.EXE.

Создан программой Sampler (версия от Feb 15 1999)
1995-98 (c) СПбГЭТУ, Мойсейчук Леонид.

Список обработанных файлов.

NN	Имя обработанного файла
1.	SHELL.CPP

Таблица с результатами измерений (используется 2 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 42	1 : 44	22033.56	1	22033.56

Исходный код файла SHELL.CPP с нумерацией строк для измерения времен выполнения ФУ представлен в приложении Г.

Результаты профилирования с измерением времен ФУ:

Список обработанных файлов.

NN	Имя обработанного файла
1.	SHELL.CPP

Таблица с результатами измерений (используется 11 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 16	1 : 18	1.68	1	1.68
1 : 18	1 : 20	10.90	7	1.56
1 : 20	1 : 23	1.68	7	0.24
1 : 23	1 : 25	43.58	27	1.61
1 : 25	1 : 33	12784.32	1849	6.91
1 : 25	1 : 28	2368.46	311	7.62
1 : 28	1 : 31	4413.42	311	14.19
1 : 31	1 : 33	0.00	311	0.00
1 : 33	1 : 25	2686.10	2133	1.26
1 : 33	1 : 35	28.50	27	1.06
1 : 35	1 : 23	24.30	20	1.22
1 : 35	1 : 37	4.19	7	0.60
1 : 37	1 : 18	5.87	6	0.98
1 : 37	1 : 39	0.84	1	0.84

По результатам профилирования видно, что время выполнения составляет примерно 22373,84 мкс. Небольшую разницу в 340,28 мкс по сравнению с измерением общего времени можно объяснить использованием эмулятора.

Можно заметить, что большое количество времени тратится на вызов функции, которая выполняет обмен местами двух элементов массива. Для улучшения программы обмен местами двух элементов массива можно перенести из вспомогательной функции в основную.

Исходный измененный код файла SHELL.CPP с нумерацией строк для измерения общего времени представлен в приложении Д.

Результаты профилирования:

```
      Список обработанных файлов.

-------------------------------------------------------------------------
   NN            Имя обработанного файла
-------------------------------------------------------------------------
   1.   SHELL.CPP
-------------------------------------------------------------------------


   Таблица с результатами измерений ( используется 2 из 416 записей )

-------------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-------------------------------------------------------------------------
 1 :  37  1 :  39           20482.24             1           20482.24
-------------------------------------------------------------------------
```

По результатам профилирования можно заметить, что после модификации программы общее время сократилось с 22033,56 мкс до 20482,24 мкс.

Исходный код измененного файла SHELL.CPP с нумерацией строк для измерения времен выполнения ФУ представлен в приложении Е.

Результаты профилирования с измерением времен ФУ:

Список обработанных файлов.

NN	Имя обработанного файла
1.	SHELL.CPP

Таблица с результатами измерений (используется 11 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 11	1 : 13	1.68	1	1.68
1 : 13	1 : 15	8.38	7	1.20
1 : 15	1 : 18	2.51	7	0.36
1 : 18	1 : 20	41.90	27	1.55
1 : 20	1 : 30	12855.56	1849	6.95
1 : 20	1 : 23	2370.98	311	7.62
1 : 23	1 : 28	2862.10	311	9.20
1 : 28	1 : 30	0.00	311	0.00
1 : 30	1 : 20	2738.06	2133	1.28
1 : 30	1 : 32	25.98	27	0.96
1 : 32	1 : 18	27.66	20	1.38
1 : 32	1 : 34	3.35	7	0.48
1 : 34	1 : 13	5.87	6	0.98
1 : 34	1 : 36	0.00	1	0.00

По результатам профилирования видно, что время выполнения составляет примерно 20844,03 мкс. Небольшую разницу по сравнению с измерением общего времени можно объяснить использованием эмулятора.

Выводы:

В результате выполнения данной лабораторной работы был изучен монитор SAMPLER, с помощью которого было выполнено профилирование тестовых программ. Также было выполнено профилирование профилирование для программы, разработанной в лабораторной работе №1. После анализа результатов удалось усовершенствовать производительность программы за счет удаления внутреннего вызова функции swap.

ПРИЛОЖЕНИЕ А.

TEST_CYC.CPP

```
1 #include <stdlib.h>
2 #include "Sampler.h"
3 #define Size 10000
4
5 int i, tmp, dim[Size];
6
7 void main()
8 {
9     SAMPLE;
10    for(i=0;i<Size/10;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
11    SAMPLE;
12    for(i=0;i<Size/5;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
13    SAMPLE;
14    for(i=0;i<Size/2;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
15    SAMPLE;
16    for(i=0;i<Size;i++)  { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
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    for(i=0;i<Size/10;i++)
31    { tmp=dim[0];
32     dim[0]=dim[i];
33     dim[i]=tmp;
34    };
35    SAMPLE;
36    for(i=0;i<Size/5;i++)
37    { tmp=dim[0];
38     dim[0]=dim[i];
39     dim[i]=tmp;
```

```
40    };
41    SAMPLE;
42    for(i=0;i<Size/2;i++)
43    { tmp=dim[0];
44     dim[0]=dim[i];
45     dim[i]=tmp;
46    };
47    SAMPLE;
48    for(i=0;i<Size;i++)
49    { tmp=dim[0];
50     dim[0]=dim[i];
51     dim[i]=tmp;
52    };
53    SAMPLE;
54 }
```

ПРИЛОЖЕНИЕ Б.

TEST_SUB.CPP

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

ПРИЛОЖЕНИЕ В.

SHELL.CPP

```
1 # include <stdio.h>
2 # include <stdlib.h>
3 # include "Sampler.h"
4
5 void swap(float* p, float* q) {
6    float hold = *p;
7    *p = *q;
8    *q = hold;
9 }
10
11 void sort(float* a, int n) {
12    int i;
13    int j;
14    int flag;
15    int jump = n;
16
17    while (jump > 0) {
18        jump = jump / 2;
19        do {
20            flag = 1;
21        for (j = 0; j < n; j++) {
22            i = j + jump;
23            if ((n>i) && (a[j] > a[i])) {
24                swap(&a[i], &a[j]);
25                flag = 0;
26            }
27        }
28        } while(!flag);
29    }
30 }
31
32 int main()
33 {
34    int n = 80;
35    float x[80];
36    int i;
37
38    for (i = 0; i < n; ++i) {
39        x[i] = (float) (rand() % 100);
```

```
40    }
41
42    SAMPLE;
43    sort(x, n);
44    SAMPLE;
45
46    return 0;
47 }
```

ПРИЛОЖЕНИЕ Г.

ФУ SHELL.CPP

```
1 # include <stdio.h>
2 # include <stdlib.h>
3 # include "Sampler.h"
4
5 void swap(float* p, float* q) {
6    float hold = *p;
7    *p = *q;
8    *q = hold;
9 }
10
11 void sort(float* a, int n) {
12    int i;
13    int j;
14    int flag;
15    int jump = n;
16    SAMPLE;
17    while (jump > 0) {
18        SAMPLE;
19        jump = jump / 2;
20        SAMPLE;
21        do {
22            flag = 1;
23        SAMPLE;
24        for (j = 0; j < n; j++) {
25            SAMPLE;
26            i = j + jump;
27            if ((n>i) && (a[j] > a[i])) {
28            SAMPLE;
29                swap(&a[i], &a[j]);
30                flag = 0;
31            SAMPLE;
32            }
33            SAMPLE;
34        }
35        SAMPLE;
36        } while(!flag);
37        SAMPLE;
38    }
39    SAMPLE;
```

```
40 }
41
42 int main()
43{
44    int n = 80;
45    float x[80];
46    int i;
47
48    for (i = 0; i < n; ++i) {
49        x[i] = (float) (rand() % 100);
50    }
51
52    sort(x, n);
53
54    return 0;
55}
```

ПРИЛОЖЕНИЕ Д.
МОДИФИЦИРОВАННЫЙ SHELL.CPP

```
1 # include <stdio.h>
2 # include <stdlib.h>
3 # include "Sampler.h"
4
5 void sort(float* a, int n) {
6    int i;
7    int j;
8    int flag;
9    int jump = n;
10    float hold=0;
11   while (jump > 0) {
12        jump = jump / 2;
13        do {
14            flag = 1;
15        for (j = 0; j < n; j++) {
16            i = j + jump;
17            if ((n>i) && (a[j] > a[i])) {
18                hold = a[i];
19                a[i] = a[j];
20                a[j] = hold;
21                flag = 0;
22            }
23        }
24        } while(!flag);
25    }
26 }
27
28 int main()
29 {
30    int n = 80;
31    float x[80];
32    int i;
33
34    for (i = 0; i < n; ++i) {
35        x[i] = (float) (rand() % 100);
36    }
37    SAMPLE;
38    sort(x, n);
```

```
39    SAMPLE;
40    return 0;
41}
```

ПРИЛОЖЕНИЕ Е.

ФУ МОДИФИЦИРОВАННОГО SHELL.CPP

```
1 # include <stdio.h>
2 # include <stdlib.h>
3 # include "Sampler.h"
4
5 void sort(float* a, int n) {
6    int i;
7    int j;
8    int flag;
9    int jump = n;
10    float hold=0;
11    SAMPLE;
12    while (jump > 0) {
13        SAMPLE;
14        jump = jump / 2;
15        SAMPLE;
16        do {
17            flag = 1;
18        SAMPLE;
19        for (j = 0; j < n; j++) {
20            SAMPLE;
21            i = j + jump;
22            if ((n>i) && (a[j] > a[i])) {
23            SAMPLE;
24                hold = a[i];
25                a[i] = a[j];
26                a[j] = hold;
27                flag = 0;
28            SAMPLE;
29            }
30            SAMPLE;
31        }
32        SAMPLE;
33        } while(!flag);
34        SAMPLE;
35    }
36    SAMPLE;
37 }
38
39 int main()
```

```
40 {
41     int n = 80;
42     float x[80];
43     int i;
44 
45     for (i = 0; i < n; ++i) {
46         x[i] = (float) (rand() % 100);
47     }
48 
49     sort(x, n);
50 
51     return 0;
52}
```