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. 7304 _____ Есиков О.И.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Изучить возможности измерения динамических характеристик программ с помощью профилировщиков на примере профилировщика SAMPLER.

Ход выполнения.

Были выполнены под управлением монитора SAMPLER тестовые программы test_cyc.cpp и test_sub.cpp. Результаты представлены на рисунках 1 и 2. Исходный код этих программ представлен в Приложении А и в Приложении Б.

Отчет о результатах измерений для программы TEST.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	4335.47	1	4335.47
1 : 11	1 : 13	8668.43	1	8668.43
1 : 13	1 : 15	21673.18	1	21673.18
1 : 15	1 : 17	43348.87	1	43348.87
1 : 17	1 : 20	4334.64	1	4334.64
1 : 20	1 : 23	8670.11	1	8670.11
1 : 23	1 : 26	21678.20	1	21678.20
1 : 26	1 : 29	43343.00	1	43343.00
1 : 29	1 : 35	4340.50	1	4340.50
1 : 35	1 : 41	8670.11	1	8670.11
1 : 41	1 : 47	21670.66	1	21670.66
1 : 47	1 : 53	43348.03	1	43348.03

Рисунок 1 – Результат работы SAMPLER для программы test_cyc.cpp

Отчет о результатах измерений для программы SUB.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	433699.02	1	433699.02
1 : 26	1 : 28	867404.75	1	867404.75
1 : 28	1 : 30	2168468.29	1	2168468.29
1 : 30	1 : 32	4336935.75	1	4336935.75

Рисунок 2 – результат работы SAMPLER для программы test_sub.cpp

Была выполнена под управлением монитора SAMPLER программа BubbleSort из лабораторной работы №1. Результат измерений для полного времени выполнения функции BubbleSort представлен на рисунке 3. Исходный код этой программы представлен в Приложении В.

Отчет о результатах измерений для программы FULL.EXE.

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

Список обработанных файлов.

NN	Имя обработанного файла
1.	BUBBLE~1.CPP

Таблица с результатами измерений (используется 2 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 51	1 : 53	67323.45	1	67323.45

Рисунок 3 – результат работы SAMPLER для измерения полного времени выполнения функции BubbleSort

Было выполнено разбиение программы BubbleSort на функциональные участки. Исходный код программы BubbleSort, разбитый на функциональные

участки, представлен в приложении Г. Полученные с помощью программы SAMPLER результаты представлены на рисунке 4.

Отчет о результатах измерений для программы PARTS.EXE.

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

Список обработанных файлов.

NN	Имя обработанного файла
1.	BUBBLE~1.CPP

Таблица с результатами измерений (используется 7 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 18	1 : 22	137.45	1	137.45
1 : 22	1 : 26	0.84	1	0.84
1 : 26	1 : 31	3459.66	75	46.13
1 : 26	1 : 37	582.48	1	582.48
1 : 31	1 : 34	23147.39	1524	15.19
1 : 34	1 : 31	20292.83	1449	14.00
1 : 34	1 : 37	21177.02	75	282.36
1 : 37	1 : 26	200.31	75	2.67
1 : 37	1 : 39	2.51	1	2.51

Рисунок 4 – Результат работы SAMPLER для измерения полного времени выполнения функции BubbleSort, разбитой на функциональные участки

В итоге общее время выполнения – 69000.49 мкс. Разницу в 1677.04 мкс с измерением для полного времени можно объяснить случайной генерацией массива для сортировки.

Как видно из результатов измерения времени выполнения функциональных участков – наиболее время затратным фрагментом является тело цикла for (строчки 29-35). Для решения этой проблемы были внесены следующие изменения:

- В 29 строчке в условии if операции с адресной арифметикой и последующим разыменованием полученного адреса были заменены на получение i-го элемента массива.
- В 32 строке операции с адресной арифметикой заменены на взятие адреса i-го элемента массива.

Была выполнена проверка изменённой программы. Результат представлен на рисунке 5. Исходный код модифицированной программы представлен в Приложении Д.

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

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


     Список обработанных файлов.

-------------------------------------------------------------------------
  NN            Имя обработанного файла
-------------------------------------------------------------------------
  1. BUBBLE~1.CPP
-------------------------------------------------------------------------

  Таблица с результатами измерений ( используется 7 из 416 записей )

-------------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-------------------------------------------------------------------------
 1 :  18  1 :  22             135.77            1             135.77
-------------------------------------------------------------------------
 1 :  22  1 :  26               0.00            1               0.00
-------------------------------------------------------------------------
 1 :  26  1 :  31            3560.23           66              53.94
 1 :  26  1 :  37             607.62            1             607.62
-------------------------------------------------------------------------
 1 :  31  1 :  34           21672.34         1510              14.35
-------------------------------------------------------------------------
 1 :  34  1 :  31           19672.64         1444              13.62
 1 :  34  1 :  37           16863.34           66             255.51
-------------------------------------------------------------------------
 1 :  37  1 :  26             127.39           66               1.93
 1 :  37  1 :  39               0.84            1               0.84
-------------------------------------------------------------------------
```

Рисунок 5 – Результат работы SAMPLER для измерения полного времени выполнения функции BubbleSort после внесения изменений

В результате внесённых изменений общее время выполнения – 62640.17 мкс.

Выводы.

В ходе выполнения лабораторной работы были изучены возможности измерения динамических характеристик программ с помощью профилировщиков на примере профилировщика SAMPLER. Для программы, взятой из первой лабораторной работы, было выполнено измерение времени работы, с последующим выявлением узких мест и их устранений – в результате чего удалось добиться более быстрого выполнения функции BubbleSort.

ПРИЛОЖЕНИЯ

Приложение А. Исходный код программы 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. Исходный код программы 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  повторений
26          SAMPLE;
27          TestLoop(Size / 5);  // 200 * 1000  повторений
28          SAMPLE;
29          TestLoop(Size / 2);  // 500 * 1000  повторений
30          SAMPLE;
31          TestLoop(Size / 1);  // 1000* 1000  повторений
32          SAMPLE;
33    }
```

Приложение B. Исходный код программы BubbleSort_full.cpp.

```
1     #include <stdio.h>
2     #include <stdlib.h>
3     #include <time.h>
4     #include <string.h>
5     #include "Sampler.h"
6
7     #define max 80
8
9     void swap(double* p, double* q)
10    {
11        double hold = *p;
12        *p = *q;
13        *q = hold;
14    }
15
16    double* sort(double* a, int n)
17    {
18        double* result = (double*)malloc(sizeof(double) * n);
19        memcpy(result, a, sizeof(double) * n);
20        char no_change;
21        do
22        {
23            no_change = 1;
24            for (int j = 0; j < n - 1; j++)
25            {
26                if (*(result + j) > *(result + j + 1))
27                {
28                    swap(result + j, result + j + 1);
29                    no_change = 0;
```

```
            }
        }
    } while (!no_change);
    return result;
}

void write_arr(double* a, int n)
{
    printf("\n");
    for (int i = 0; i < n; i++)
        printf("%7.1f ", *(a + i));
    printf("\n");
}

int main()
{
    int n = max;
    double x[max];
    srand(time(NULL));
    for (int i = 0; i < n; i++)
        x[i] = rand() % 100;
    SAMPLE;
    sort(x, n);
    SAMPLE;
    return 0;
}
```

Приложение Г. Исходный код программы BubbleSort_parts.cpp.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include "Sampler.h"

#define max 80

void swap(double* p, double* q)
{
    double hold = *p;
    *p = *q;
    *q = hold;
}

double* sort(double* a, int n)
{
    SAMPLE;
    double* result = (double*)malloc(sizeof(double) * n);
    memcpy(result, a, sizeof(double) * n);
    char no_change;
    SAMPLE;
    do
    {
        no_change = 1;
        SAMPLE;
        for (int j = 0; j < n - 1; j++)
        {
            if (*(result + j) > *(result + j + 1))
            {
                SAMPLE;
                swap(result + j, result + j + 1);
                no_change = 0;
                SAMPLE;
```

```
35                  }
36              }
37              SAMPLE;
38          } while (!no_change);
39          SAMPLE;
40          return result;
41      }
42
43      void write_arr(double* a, int n)
44      {
45          printf("\n");
46          for (int i = 0; i < n; i++)
47              printf("%7.1f ", *(a + i));
48          printf("\n");
49      }
50
51      int main()
52      {
53          int n = max;
54          double x[max];
55          srand(time(NULL));
56          for (int i = 0; i < n; i++)
57              x[i] = rand() % 100;
58          sort(x, n);
59          return 0;
60      }
```

Приложение Д. Исходный код программы BubbleSort_updates.cpp.

```
1       #include <stdio.h>
2       #include <stdlib.h>
3       #include <time.h>
4       #include <string.h>
5       #include "Sampler.h"
6
7       #define max 80
8
9       void swap(double* p, double* q)
10      {
11          double hold = *p;
12          *p = *q;
13          *q = hold;
14      }
15
16      double* sort(double* a, int n)
17      {
18          SAMPLE;
19          double* result = (double*)malloc(sizeof(double) * n);
20          memcpy(result, a, sizeof(double) * n);
21          char no_change;
22          SAMPLE;
23          do
24          {
25              no_change = 1;
26              SAMPLE;
27              for (int j = 0; j < n - 1; j++)
28              {
29                  if (result[j] > result[j + 1])
30                  {
31                      SAMPLE;
32                      swap(&result[j], &result[j + 1]);
33                      no_change = 0;
34                      SAMPLE;
```

```
35                  }
36              }
37              SAMPLE;
38          } while (!no_change);
39          SAMPLE;
40          return result;
41      }
42
43      void write_arr(double* a, int n)
44      {
45          printf("\n");
46          for (int i = 0; i < n; i++)
47              printf("%7.1f ", *(a + i));
48          printf("\n");
49      }
50
51      int main()
52      {
53          int n = max;
54          double x[max];
55          srand(time(NULL));
56          for (int i = 0; i < n; i++)
57              x[i] = rand() % 100;
58          sort(x, n);
59          return 0;
60      }
```