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 _____ Субботин А.С.

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:
   a. измерение только полного времени выполнения программы;
   b. 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.

Исходные данные

Вариант 17. Исходный код для пунктов 2-3 задания был получен преобразованием исходного кода на Си из лабораторной работы №1 соответствующего варианта.

Ход работы

Для использования была выбрана старая версия монитора Sampler_old, соответственно запуск производился под эмулятором DOSBox-0.74

1. Произведено профилирование тестового файла TEST_CYC.EXE. Исходный код программы представлен в Приложении А. Результаты профилирования представлены на Рисунке 1.

```
-----------------------------------------------------------------------
   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            4337.15            1            4337.15
-----------------------------------------------------------------------
 1 :  11  1 :  13            8670.11            1            8670.11
-----------------------------------------------------------------------
 1 :  13  1 :  15           21679.04            1           21679.04
-----------------------------------------------------------------------
 1 :  15  1 :  17           43343.84            1           43343.84
-----------------------------------------------------------------------
 1 :  17  1 :  20            4341.34            1            4341.34
-----------------------------------------------------------------------
 1 :  20  1 :  23            8670.95            1            8670.95
-----------------------------------------------------------------------
 1 :  23  1 :  26           21672.34            1           21672.34
-----------------------------------------------------------------------
 1 :  26  1 :  29           43349.70            1           43349.70
-----------------------------------------------------------------------
 1 :  29  1 :  35            4336.31            1            4336.31
-----------------------------------------------------------------------
 1 :  35  1 :  41            8670.11            1            8670.11
-----------------------------------------------------------------------
 1 :  41  1 :  47           21678.20            1           21678.20
-----------------------------------------------------------------------
 1 :  47  1 :  53           43343.00            1           43343.00
-----------------------------------------------------------------------
```

Рисунок 1 – Результат профилирования TEST_CYC.EXE

Участок 9-11 в 10 раз меньше, 11-13 в 5 раз меньше, 13-15 в 2 раза меньше участка 15-17, что согласуется с соответственной кратной разницей в количестве итераций циклов.

2. Произведено профилирование тестового файла TEST_SUB.EXE. Исходный код программы представлен в Приложении Б. Результаты профилирования представлены на Рисунке 2.

```
-------------------------------------------------------------------------
   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          867392.18             1          867392.18
-------------------------------------------------------------------------
 1 :  28  1 :  30         2168481.70             1         2168481.70
-------------------------------------------------------------------------
 1 :  30  1 :  32         4336951.68             1         4336951.68
-------------------------------------------------------------------------
```

Рисунок 2 – Результат профилирования TEST_SUB.EXE

Как и в пункте 1, затраченное время на выполнение функции уменьшается столько же кратно, сколь кратно уменьшается количество вызовов этой функции (в 2, в 5 и в 10 раз).

3. Был взят код, написанный в ходе выполнения ЛР №1. Для того, чтобы команда компилировалась, была произведена замена «#include <stdbool.h>» на «typedef enum { false, true } bool;», т.к. изначальный файл был добавлен в более поздние версии языка Си.

4. Полный замер (код в Приложении В) представлен на Рисунке 3.

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

   Таблица с результатами измерений ( используется 2 из 416 записей )

-----------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-----------------------------------------------------------------------
 1 :  39  1 :  60            3283.66            1            3283.66
-----------------------------------------------------------------------
```

Рисунок 3 – Измерение полного времени выполнения программы

5. Отдельные замеры (код в Приложении Г) представлен на Рисунке 4.

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

   Таблица с результатами измерений ( используется 12 из 416 записей )

-----------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-----------------------------------------------------------------------
 1 :  40  1 :  46              53.64            1              53.64
-----------------------------------------------------------------------
 1 :  46  1 :  62             246.40            1             246.40
 1 :  46  1 :  57              14.25            1              14.25
 1 :  46  1 :  52               9.22            1               9.22
 1 :  46  1 :  48               5.03            1               5.03
-----------------------------------------------------------------------
 1 :  48  1 :  50               0.84            1               0.84
-----------------------------------------------------------------------
 1 :  50  1 :  68               4.19            1               4.19
-----------------------------------------------------------------------
 1 :  52  1 :  55              87.16            1              87.16
-----------------------------------------------------------------------
 1 :  55  1 :  68               4.19            1               4.19
-----------------------------------------------------------------------
 1 :  57  1 :  60            1240.38            1            1240.38
-----------------------------------------------------------------------
 1 :  60  1 :  68               4.19            1               4.19
-----------------------------------------------------------------------
 1 :  62  1 :  65            1535.39            1            1535.39
-----------------------------------------------------------------------
 1 :  65  1 :  68              87.16            1              87.16
-----------------------------------------------------------------------
 1 :  68  1 :  46               5.03            3               1.68
 1 :  68  1 :  70               1.68            1               1.68
-----------------------------------------------------------------------
```

Рисунок 4 – Измерение времен выполнения ФУ

53.64 + 246.4 + 14.25 + 9.22 + 5.03 + 0.84 + 4.19 + 87.16 + 4.19 + 1240.38 + 4.19 +1535.39 + 87.16 + 5.03 + 1.68 = 3298.75 – Суммарное время отдельных замеров.

Время выполнения программы с отдельными замерами (3298.75 мкс) больше полного (3283.66 мкс) на 15 мкс (0,5%).

6. Из-за невозможности улучшения реализации вычислительных функций возможно лишь незначительное улучшение производительности. Изменения:
   - Избавление от переменной «done»
   - Замена цикла do while на for
   - (потенциально) Если бы Turbo C++ 3.0 поддерживал стандарт C99, то сами функции ошибок можно было бы отдать библиотеке math.h, а не считать разложением в ряд

   Модифицированный код представлен в приложении Д.
7. Измерение полного времени выполнения модифицированной программы представлено на Рисунке 5.

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

   Таблица с результатами измерений ( используется 2 из 416 записей )

-------------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-------------------------------------------------------------------------
 1 :  39  1 :  53            3240.08            1            3240.08
-------------------------------------------------------------------------
```

Рисунок 5 – Измерение полного времени выполнения модифицированной программы

С модифицированным кодом удалось выиграть около 43 мкс.

Выводы

В ходе выполнения данной лабораторной работы было произведено вычисление профиля программы на Си с помощью профилировщика Sampler. Произведена оптимизация программы из ЛР1.

Приложение A. 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  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 В. Код программы из ЛР №1 для проведения замера полного времени выполнения программы

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "sampler.h"
typedef enum {false, true} bool;
double erf(double x){
    const double sqrtpi       = 1.7724538;
    double t2           = 0.66666667;
    double t3           = 0.66666667;
    double t4           = 0.07619048;
    double t5           = 0.01693122;
    double t6           = 3.078403E-3;
    double t7           = 4.736005E-4;
    double t8           = 6.314673E-5;
    double t9           = 7.429027E-6;
    double t10          = 7.820028E-7;
    double t11          = 7.447646E-8;
    double t12          = 6.476214E-9;

    double x2, sum;
    x2 = x*x;
    sum = t5+x2*(t6+x2*(t7+x2*(t8+x2*(t9+x2*(t10+x2*(t11+x2*t12))))));
    return (2.0*exp(-x2)/sqrtpi*(x*(1+x2*(t2+x2*(t3+x2*(t4+x2*sum))))));

    }

    double erfc(double x){
          const double sqrtpi = 1.7724538;
          double x2,v,sum;
          x2 = x*x;
          v = 1/(2*x2);
          sum=v/(1+8*v/(1+9*v/(1+10*v/(1+11*v/(1+12*v)))));
          sum=v/(1+3*v/(1+4*v/(1+5*v/(1+6*v/(1+7*sum))))));
          return (1.0/(exp(x2)*x*sqrtpi*(1+v/(1+2*sum))));
     }

     int main()
     {
          SAMPLE;
          double x,er,ec;
          bool done;
          x = 2.0;
          done = false;
          do{
              if(x<0){
                  done = true;
            }else if (x == 0){
                      er = 0;
                ec = 1;
                  }else if (x < 1.5){
                er = erf(x);
                ec = 1 - er;
            }else{
                ec = erfc(x);
                er = 1-ec;
                  }
            x = x - 1;

          }while (done == false);
```

```
60.           SAMPLE;
61.           return 0;
62.       }
```

Приложение Г. Код программы из ЛР №1 для измерения времен выполнения ФУ

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "SAMPLER.H"
typedef enum { false, true } bool;

double erf(double x){
    const double sqrtpi       = 1.7724538;
    double t2           = 0.66666667;
    double t3           = 0.66666667;
    double t4           = 0.07619048;
    double t5           = 0.01693122;
    double t6           = 3.078403E-3;
    double t7           = 4.736005E-4;
    double t8           = 6.314673E-5;
    double t9           = 7.429027E-6;
    double t10          = 7.820028E-7;
    double t11          = 7.447646E-8;
    double t12          = 6.476214E-9;

    double x2, sum;
    x2 = x*x;
    sum = t5+x2*(t6+x2*(t7+x2*(t8+x2*(t9+x2*(t10+x2*(t11+x2*t12))))));
    return (2.0*exp(-x2)/sqrtpi*(x*(1+x2*(t2+x2*(t3+x2*(t4+x2*sum))))));

     }

     double erfc(double x){
         const double sqrtpi = 1.7724538;
         double x2,v,sum;
         x2 = x*x;
         v = 1/(2*x2);
         sum=v/(1+8*v/(1+9*v/(1+10*v/(1+11*v/(1+12*v)))));
         sum=v/(1+3*v/(1+4*v/(1+5*v/(1+6*v/(1+7*sum))))));
         return (1.0/(exp(x2)*x*sqrtpi*(1+v/(1+2*sum))));
     }

     int main()
     {
         SAMPLE;
         double x,er,ec;
         bool done;
         x = 2.0;
         done = false;
         do{
                 SAMPLE;
             if(x<0){
                      SAMPLE;
                 done = true;
                      SAMPLE;
             }else if (x == 0){
                      SAMPLE;
                 er = 0;
                 ec = 1;
                      SAMPLE;
             }else if (x < 1.5){
                      SAMPLE;
                 er = erf(x);
                 ec = 1 - er;
```

```
60.                         SAMPLE;
61.             }else{
62.                         SAMPLE;
63.                 ec = erfc(x);
64.                 er = 1-ec;
65.                         SAMPLE;
66.             }
67.             x = x - 1;
68.                 SAMPLE;
69.         }while (done == false);
70.         SAMPLE;
71.         return 0;
72.     }
```

Приложение Д. Модифицированный код программы из ЛР №1 для проведения замера полного времени выполнения программы

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "SAMPLER.H"

double erf(double x){
    const double sqrtpi       = 1.7724538;
    double t2           = 0.66666667;
    double t3           = 0.66666667;
    double t4           = 0.07619048;
    double t5           = 0.01693122;
    double t6           = 3.078403E-3;
    double t7           = 4.736005E-4;
    double t8           = 6.314673E-5;
    double t9           = 7.429027E-6;
    double t10          = 7.820028E-7;
    double t11          = 7.447646E-8;
    double t12          = 6.476214E-9;

    double x2, sum;
    x2 = x*x;
    sum = t5+x2*(t6+x2*(t7+x2*(t8+x2*(t9+x2*(t10+x2*(t11+x2*t12))))));
    return (2.0*exp(-x2)/sqrtpi*(x*(1+x2*(t2+x2*(t3+x2*(t4+x2*sum))))));

     }

     double erfc(double x){
         const double sqrtpi = 1.7724538;
         double x2,v,sum;
         x2 = x*x;
         v = 1/(2*x2);
         sum=v/(1+8*v/(1+9*v/(1+10*v/(1+11*v/(1+12*v)))));
         sum=v/(1+3*v/(1+4*v/(1+5*v/(1+6*v/(1+7*sum))))));
         return (1.0/(exp(x2)*x*sqrtpi*(1+v/(1+2*sum))));
     }

     int main()
     {
         SAMPLE;
         double x,er,ec;
         for(x = 0; x<=2.0; x++){
           if (x == 0){
               er = 0;
               ec = 1;
            }else if (x < 1.5){
               er = erf(x);
               ec = 1 - er;
           }else{
               ec = erfc(x);
               er = 1-ec;
                 }
         }
         SAMPLE;
         return 0;
     }
```