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. 6304 Блинникова Ю. И.

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 С, 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

Использовался старый SAMPLER. Программы компилировались с помощью Borland C++. Компилирование выполнялось на Windows XP, профилирование – в DOSBox.

Тестовая программа test_cyc.cpp

Код программы test_cyc.cpp с нумерацией строк представлен в приложении А.

Результаты профилирования:

```
-----------------------------------------------------------------------
   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 :   8  1 :  10            4335.47            1            4335.47
-----------------------------------------------------------------------
 1 :  10  1 :  12            8675.98            1            8675.98
-----------------------------------------------------------------------
 1 :  12  1 :  14           21671.50            1           21671.50
-----------------------------------------------------------------------
 1 :  14  1 :  16           43348.87            1           43348.87
-----------------------------------------------------------------------
 1 :  16  1 :  19            4337.15            1            4337.15
-----------------------------------------------------------------------
 1 :  19  1 :  22            8668.43            1            8668.43
-----------------------------------------------------------------------
 1 :  22  1 :  25           21672.34            1           21672.34
-----------------------------------------------------------------------
 1 :  25  1 :  28           43348.03            1           43348.03
-----------------------------------------------------------------------
 1 :  28  1 :  34            4334.64            1            4334.64
-----------------------------------------------------------------------
 1 :  34  1 :  40            8670.11            1            8670.11
-----------------------------------------------------------------------
 1 :  40  1 :  46           21676.53            1           21676.53
-----------------------------------------------------------------------
 1 :  46  1 :  52           43348.87            1           43348.87
```

По результатам профилирования тестовых программ можно сделать выводы о том, что время выполнения линейно зависит от количества итераций цикла, а также время сильно завышено из-за накладных затрат эмулятора.

Тестовая программа test_sub.cpp

Код программы test_sub.cpp с нумерацией строк представлен в приложении В.

Результаты профилирования:

```
-----------------------------------------------------------------------
   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 :  30  1 :  32           433699.86             1          433699.86
-----------------------------------------------------------------------
 1 :  32  1 :  34           867392.18             1          867392.18
-----------------------------------------------------------------------
 1 :  34  1 :  36          2168480.87             1         2168480.87
-----------------------------------------------------------------------
 1 :  36  1 :  38          4336949.16             1         4336949.16
-----------------------------------------------------------------------
```

По результатам профилирования можно сделать выводы аналогично тестовой программе test_cyc.cpp, что время выполнения линейно зависит от количества итераций цикла, а также время сильно завышено из-за накладных затрат эмулятора.

Программа из первой лабораторной работы l1full.cpp (для измерения полного времени).

Код программы из первой лабораторной работы с нумерацией строк представлен в приложении С.

Результаты профилирования с измерением полного времени:

```
-----------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-----------------------------------------------------------------------
 1 :  49  1 :  51           245744.98             1          245744.98
-----------------------------------------------------------------------
```

Общее время выполнения – 245744.98 мкс. Результаты также завышены из-за накладных затрат эмулятора.

Программа из первой лабораторной работы l1.cpp (для измерения времен выполнения ФУ).

Код программы из первой лабораторной работы с нумерацией строк представлен в приложении D.

Результаты профилирования с измерением времен выполнения ФУ:

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 27	1 : 34	243.05	1	243.05
1 : 34	1 : 39	243120.90	11	22101.90
1 : 39	1 : 34	1004.04	10	100.40
1 : 39	1 : 44	42.74	1	42.74

По результатам измерений времени на ФУ видно, что время выполнения функции – 244410,73 мкс.

Программа из первой лабораторной работы l1.cpp (исследование каждого цикла в for).

Код программы из первой лабораторной работы с нумерацией строк представлен в приложении E и F.

Результаты профилирования с измерением времен выполнения:

Таблица с результатами измерений (используется 5 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 27	1 : 34	243.89	1	243.89
1 : 34	1 : 36	39.39	11	3.58
1 : 36	1 : 36	245594.13	4083	60.15
1 : 36	1 : 40	656.23	11	59.66
1 : 40	1 : 34	1014.10	10	101.41
1 : 40	1 : 45	43.58	1	43.58

По результатам измерений времени видно, что время выполнения функции – 247591,32 мкс.

Результаты профилирования с измерением времен выполнения измененной программы:

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 27	1 : 34	243.89	1	243.89
1 : 34	1 : 36	35.20	11	3.20
1 : 36	1 : 36	226545.03	4083	55.48
1 : 36	1 : 40	601.75	11	54.70
1 : 40	1 : 34	1011.58	10	101.16
1 : 40	1 : 45	43.58	1	43.58

По результатам измерений времени видно, что время выполнения функции изменённой программы меньше исходной и составляет 228481,03 мкс.

Выводы

В результате выполнения данной лабораторной работы был изучен монитор SAMPLER, с помощью которого было выполнено профилирование тестовых программ test_cyc.cpp и test_sub.cpp.

Было проанализировано полное время выполнения программы, разработанной в 1-ой лабораторной работе, и время выполнения её ФУ.

ПРИЛОЖЕНИЕ А

TEST_CYC.C

```
#include <stdlib.h>
#include "Sampler.h"
#define Size 10000
int i, tmp, dim[Size];

void main()
{
    SAMPLE;
    for(i=0;i<Size/10;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
    SAMPLE;
    for(i=0;i<Size/5;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
    SAMPLE;
    for(i=0;i<Size/2;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
    SAMPLE;
    for(i=0;i<Size;i++)  { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
    SAMPLE;
    for(i=0;i<Size/10;i++)
      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
    SAMPLE;
    for(i=0;i<Size/5;i++)
      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
    SAMPLE;
   for(i=0;i<Size/2;i++)
      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
    SAMPLE;
    for(i=0;i<Size;i++)
      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
    SAMPLE;
    for(i=0;i<Size/10;i++)
      { tmp=dim[0];
        dim[0]=dim[i];
        dim[i]=tmp;
      };
    SAMPLE;
    for(i=0;i<Size/5;i++)
     { tmp=dim[0];
       dim[0]=dim[i];
        dim[i]=tmp;
      };
    SAMPLE;
    for(i=0;i<Size/2;i++)
      { tmp=dim[0];
        dim[0]=dim[i];
        dim[i]=tmp;
      };
    SAMPLE;
    for(i=0;i<Size;i++)
      { tmp=dim[0];
        dim[0]=dim[i];
        dim[i]=tmp;
      };
    SAMPLE;
}
```

ПРИЛОЖЕНИЕ В

TEST_SUB.C

```
#include <stdlib.h>
#include "Sample.h"
const unsigned Size = 1000;


void TestLoop(int nTimes)
{
   static int TestDim[Size];
   int tmp;
   int iLoop;

   while (nTimes > 0)
   {
    nTimes --;

    iLoop = Size;
    while (iLoop > 0)
   {
      iLoop -- ;
      tmp = TestDim[0];
      TestDim[0] = TestDim[nTimes];
      TestDim[nTimes] = tmp;
    }
   }
 } /* TestLoop */


 void main()
 {
   SAMPLE;
   TestLoop(Size / 10);
   SAMPLE;
   TestLoop(Size / 5);
   SAMPLE;
   TestLoop(Size / 2);
   SAMPLE;
   TestLoop(Size / 1);
   SAMPLE;
 }

```

ПРИЛОЖЕНИЕ С

Полное время L1FULL.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include "math.h"
#include "Sampler.h"

const double tol= 1.0E-6;

double fx(double x){
    return exp(-1.0*x/2.0);
}

double dfx(double x){
    return (-1.0*exp(-1.0*x/2.0)/2.0);
}

double simps(double lower,double upper,double tol,double sum){
    double x,delta_x,even_sum,odd_sum,end_sum,end_cor,suml;
    int pieces;

    pieces=2;
    delta_x=(upper-lower)/pieces;
    odd_sum=fx(lower+delta_x);
    even_sum=0.0;
    end_sum=fx(lower)+fx(upper);
    end_cor=dfx(lower)-dfx(upper);
    sum=(end_sum+4.0*odd_sum)*delta_x/3.0;
    do{
        pieces=pieces*2;
        suml=sum;
        delta_x=(upper-lower)/pieces;
        even_sum=even_sum+odd_sum;
        odd_sum=0.0;
        for (int i=1;i<=pieces/2;i++){
            x=lower+delta_x*(2.0*i-1.0);
            odd_sum=odd_sum+fx(x);
        }
         sum=(7.0*end_sum+14.0*even_sum+16.00*odd_sum
                    +end_cor*delta_x)*delta_x/15.0;
    }
    while ((sum!=suml) & (abs(sum-suml)<=abs(tol*sum)));
    return sum;
}

int main(void) {
    double  sum=0.0;
    double lower=1.0;
    double upper=9.0;
    double res =0.0;
    SAMPLE;
    simps(lower,upper,tol,sum);
    SAMPLE;
    return 0;
}
```

ПРИЛОЖЕНИЕ D

Время ФУ L1.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include "math.h"
#include "Sampler.h"

const double tol= 1.0E-6;

double fx(double x){
    return exp(-1.0*x/2.0);
}

double dfx(double x){
    return (-1.0*exp(-1.0*x/2.0)/2.0);
}

double simps(double lower,double upper,double tol,double sum){
    double x,delta_x,even_sum,odd_sum,end_sum,end_cor,suml;
    int pieces;

    pieces=2;
    delta_x=(upper-lower)/pieces;
    odd_sum=fx(lower+delta_x);
    even_sum=0.0;
    end_sum=fx(lower)+fx(upper);
    end_cor=dfx(lower)-dfx(upper);
    sum=(end_sum+4.0*odd_sum)*delta_x/3.0;
    SAMPLE;
    do{
        pieces=pieces*2;
        suml=sum;
        delta_x=(upper-lower)/pieces;
        even_sum=even_sum+odd_sum;
        odd_sum=0.0;
        SAMPLE;
        for (int i=1;i<=pieces/2;i++){
            x=lower+delta_x*(2.0*i-1.0);
            odd_sum=odd_sum+fx(x);
        }
        SAMPLE;
        sum=(7.0*end_sum+14.0*even_sum+16.00*odd_sum
                +end_cor*delta_x)*delta_x/15.0;
    }
    while ((sum!=suml) & (abs(sum-suml)<=abs(tol*sum)));
    SAMPLE;
    return sum;
}

int main(void) {
    double  sum=0.0;
    double lower=1.0;
    double upper=9.0;
    double res =0.0;
    simps(lower,upper,tol,sum);
    return 0;
}
```

ПРИЛОЖЕНИЕ Е

Исследование цикла for L1.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include "math.h"
#include "Sampler.h"

const double tol= 1.0E-6;

double fx(double x){
    return exp(-1.0*x/2.0);
}

double dfx(double x){
    return (-1.0*exp(-1.0*x/2.0)/2.0);
}

double simps(double lower,double upper,double tol,double sum){
    double x,delta_x,even_sum,odd_sum,end_sum,end_cor,suml;
    int pieces;

    pieces=2;
    delta_x=(upper-lower)/pieces;
    odd_sum=fx(lower+delta_x);
    even_sum=0.0;
    end_sum=fx(lower)+fx(upper);
    end_cor=dfx(lower)-dfx(upper);
    sum=(end_sum+4.0*odd_sum)*delta_x/3.0;
    SAMPLE;
    do{
        pieces=pieces*2;
        suml=sum;
        delta_x=(upper-lower)/pieces;
        even_sum=even_sum+odd_sum;
        odd_sum=0.0;
        SAMPLE;
        for (int i=1;i<=pieces/2;i++){
            SAMPLE;
            x=lower+delta_x*(2.0*i-1.0);

            odd_sum=odd_sum+fx(x);
        }
        SAMPLE;
        sum=(7.0*end_sum+14.0*even_sum+16.00*odd_sum
                +end_cor*delta_x)*delta_x/15.0;
    }
    while ((sum!=suml) & (abs(sum-suml)<=abs(tol*sum)));
    SAMPLE;
    return sum;
}

int main(void) {
    double  sum=0.0;
    double lower=1.0;
    double upper=9.0;
    double res =0.0;
    simps(lower,upper,tol,sum);
    return 0;
}
```

ПРИЛОЖЕНИЕ F

Измененная программа L1.CPP

```
#include <stdio.h>
#include <stdlib.h>
#include "math.h"
#include "Sampler.h"

const double tol= 1.0E-6;

double fx(double x){
    return exp(-1.0*x/2.0);
}

double dfx(double x){
    return (-1.0*exp(-1.0*x/2.0)/2.0);
}

double simps(double lower,double upper,double tol,double sum){
    double x,delta_x,even_sum,odd_sum,end_sum,end_cor,suml;
    int pieces;

    pieces=2;
    delta_x=(upper-lower)/pieces;
    odd_sum=fx(lower+delta_x);
    even_sum=0.0;
    end_sum=fx(lower)+fx(upper);
    end_cor=dfx(lower)-dfx(upper);
    sum=(end_sum+4.0*odd_sum)*delta_x/3.0;
    SAMPLE;
    do{
        pieces=pieces*2;
        suml=sum;
        delta_x=(upper-lower)/pieces;
        even_sum=even_sum+odd_sum;
        odd_sum=0.0;
        SAMPLE;
        for (int i=1;i<=pieces/2;i++){
            SAMPLE;
            x=lower+delta_x*(2.0*i-1.0);
            odd_sum=odd_sum+exp(-1.0*x/2.0);
        }
        SAMPLE;
        sum=(7.0*end_sum+14.0*even_sum+16.00*odd_sum
            +end_cor*delta_x)*delta_x/15.0;
    }
    while ((sum!=suml) & (abs(sum-suml)<=abs(tol*sum)));
    SAMPLE;
    return sum;
}

int main(void) {
    double  sum=0.0;
    double lower=1.0;
    double upper=9.0;
    double res =0.0;
    simps(lower,upper,tol,sum);
    return 0;
}
```