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 _____ Петруненко Д.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

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

Постановка задачи.

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);
3. Выявить "узкие места", связанные с ухудшением производительности программы, ввести в программу усовершенствования и получить новые профили. Объяснить смысл введенных модификаций программ.

Ход выполнения.

1. Файл TEST_CYC.CPP был дополнен контрольными точками и с компилирован с помощью Borland C++. Исходный код TEST_CYC.CPP в приложении А. Затем исполняемый файл был подан на вход программе SAMPLER.exe. Результат работы программы SAMPLER:

```
    Отчет о результатах измерений для программы TEST_CYC.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            4337.15            1            4337.15
-------------------------------------------------------------------------
 1 :  11  1 :  13            8669.27            1            8669.27
-------------------------------------------------------------------------
 1 :  13  1 :  15           21678.20            1           21678.20
-------------------------------------------------------------------------
 1 :  15  1 :  17           43343.84            1           43343.84
-------------------------------------------------------------------------
 1 :  17  1 :  20            4341.34            1            4341.34
-------------------------------------------------------------------------
 1 :  20  1 :  23            8671.78            1            8671.78
-------------------------------------------------------------------------
 1 :  23  1 :  26           21672.34            1           21672.34
-------------------------------------------------------------------------
 1 :  26  1 :  29           43349.70            1           43349.70
-------------------------------------------------------------------------
 1 :  29  1 :  35            4337.15            1            4337.15
-------------------------------------------------------------------------
 1 :  35  1 :  41            8669.27            1            8669.27
-------------------------------------------------------------------------
 1 :  41  1 :  47           21679.04            1           21679.04
-------------------------------------------------------------------------
 1 :  47  1 :  53           43343.84            1           43343.84
-------------------------------------------------------------------------
```

Рис. 1: Результаты профилирования тестовой программы test_cyc.cpp

2. Файл TEST_SUB.CPP был дополнен контрольными точками и с компилирован с помощью Borland C++. Исходный код TEST_SUB.CPP в

приложении Б. Затем исполняемый файл был подан на вход программе SAMPLER.exe. Результат работы программы SAMPLER:

```
Отчет о результатах измерений для программы TEST_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 :  28  1 :  30          433699.02            1          433699.02
-------------------------------------------------------------------------
 1 :  30  1 :  32          867385.47            1          867385.47
-------------------------------------------------------------------------
 1 :  32  1 :  34         2168498.47            1         2168498.47
-------------------------------------------------------------------------
 1 :  34  1 :  36         4337000.28            1         4337000.28
-------------------------------------------------------------------------
```

Таблица 2: Результаты профилирования тестовой программы test_sub.cpp

3. Были расставлены контрольные точки для исходного кода первой лабораторной работы в файл romb.cpp и скомпилирован с помощью Borland C++. Исходный код romb.cpp в приложении В. Затем исполняемый файл был подан на вход программе SAMPLER.exe и получено измерение полного времени выполнения программы. Результат работы программы SAMPLER:

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

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



        Список обработанных файлов.

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


   Таблица с результатами измерений ( используется 2 из 416 записей )

-------------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-------------------------------------------------------------------------
 1 :  76  1 :  78             6151.63             1            6151.63
-------------------------------------------------------------------------
```

Рис 3: Результаты профилирования программы из первой лабораторной работы (полное время работы программы)

4. Контрольные точки были расставлены на все функциональные участки программы для исходного кода первой лабораторной работы в файл romb.cpp и скомпилирован с помощью Borland C++. Исходный код romb.cpp в приложении Г. Затем исполняемый файл был подан на вход программе SAMPLER.exe и получено измерение всех функциональных участков во время выполнения программы. Результат работы программы SAMPLER:

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

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



        Список обработанных файлов.

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


   Таблица с результатами измерений ( используется 7 из 416 записей )

-------------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-------------------------------------------------------------------------
 1 :  14  1 :  35            1170.82            1            1170.82
-------------------------------------------------------------------------
 1 :  35  1 :  43             160.91            1             160.91
-------------------------------------------------------------------------
 1 :  43  1 :  53            1983.77            4             495.94
-------------------------------------------------------------------------
 1 :  53  1 :  60            2293.87            4             573.47
-------------------------------------------------------------------------
 1 :  60  1 :  73             369.60            4              92.40
-------------------------------------------------------------------------
 1 :  73  1 :  43              40.23            3              13.41
 1 :  73  1 :  76              55.31            1              55.31
-------------------------------------------------------------------------
```

Таблица 4: Результаты профилирования программы из первой лабораторной работы (разбитие на функциональные участки)

По результатам профилирования видно, что примерное время выполнения составляет 6074,51 мкс. При этом наибольшее время выполнения получается на функциональных участках итеративного инициализации переменных, а также в двух основных циклах выполняющих вычисления. Для улучшения показателей на функциональном участке итеративный инициализацией возможно заменить более быстрым обнулением массива при инициализации {0}. Для первого основного цикла с вычислениями возможно заменить вызов дополнительной функции тем же пересчётом в основной

функции, а также убрать избыточную переменную i из тела цикла. Для второго основного цикла с вычислениями возможно убрать избыточную переменную nt из тела цикла. Исходный код с изменениями представлен в приложении Д. Результат работы программы SAMPLER:

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

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



        Список обработанных файлов.

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


   Таблица с результатами измерений ( используется 7 из 416 записей )

-------------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-------------------------------------------------------------------------
 1 :  14  1 :  29             626.06             1             626.06
-------------------------------------------------------------------------
 1 :  29  1 :  37             160.91             1             160.91
-------------------------------------------------------------------------
 1 :  37  1 :  46            1887.39             4             471.85
-------------------------------------------------------------------------
 1 :  46  1 :  52            2283.81             4             570.95
-------------------------------------------------------------------------
 1 :  52  1 :  65             368.76             4              92.19
-------------------------------------------------------------------------
 1 :  65  1 :  37              38.55             3              12.85
 1 :  65  1 :  68              54.48             1              54.48
-------------------------------------------------------------------------
```

Рис 5: Результаты профилирования изменённой программы из первой лабораторной работы

По результатам профилирования видно, что примерное время выполнения составляет 5319,96 мкс. Уменьшение времени работы составило 654,55 мкс (~ 11%).

Выводы.

В ходе выполнения лабораторной работы был изучен монитор SAMOLER, который позволяет выполнять профилирование программ. Было выполнено профилирование программ для TEST_CYC.CPP, TEST_SUB.CPP, а также для программы первой лабораторной работы. В ходи профилирования удалось обнаружить проблемные участки кода, оптимизация которых позволила существенно увеличить скорость работы программы, примерно на 655 мкс, около 11%.

ПРИЛОЖЕНИЕ А.

TEST_CYC.CPP

```
#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 Б.

TEST_SUB.CPP

```
#include "sampler.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); // 100 * 1000  повторений
   SAMPLE ;
   TestLoop(Size / 5);  // 200 * 1000  повторений
   SAMPLE ;
   TestLoop(Size / 2);  // 500 * 1000  повторений
   SAMPLE ;
   TestLoop(Size / 1);  // 1000* 1000  повторений
   SAMPLE ;  }
```

ПРИЛОЖЕНИЕ В.

ROMB.CPP

```
#include <stdio.h>
#include <math.h>
#include "sampler.h"

double tol = 0.0001;
int done = 0;
double sumMain,upper,lower;

double fx(double x) {
   return 1.0/x;
}

void romb(double lower,double upper){
   int p;
   int nx[16];
   for (p = 0; p < 16; p++) {
      nx[p] = 0;
   }
   double t[136];
   for (p = 0; p < 136; p++) {
      t[p] = 0.0;
   }
   int done,error;
   int pieces,nt,i,ii,n,nn,l,ntra,k,m,j;
   double delta_x,c,sum,fotom,x;
   pieces = 1;
   nx[0] = 1;
   delta_x = (upper-lower)/pieces;
```

```
c = (fx(lower)+fx(upper))*0.5;
t[0] = delta_x*c;
n = 1;
nn = 2;
sum = c;
do {
   n = n+1;
   fotom = 4.0;
   nx[n-1] =  nn;
   pieces = pieces*2;
   l = pieces-1;
   delta_x = (upper-lower)/pieces;
   for (ii =1; ii<=((l+1)/2); ii++) {
      i = ii*2-1;
      x = lower+i *delta_x;
      sum = sum+fx(x);
   }
   t[nn-1] = delta_x *sum;
   printf("%d%f ", pieces,t[nn]);
   ntra = nx[n-2];
   k = n-1;
   for (m = 1;m<=k;m++) {
      j = nn + m;
      nt = nx[n-1-1]+m-1;
      t[j-1] = (fotom *t[j - 2]-t[nt-1])/(fotom-1.0);
   }
   printf("%d%f\n",j-1,t[j-1]);
   if (n>4) {
      if (t[nn] < 0.0 || t[nn] > 0.0) {
         if ((abs(t[ntra + 1]-t[nn-1])<=abs(t[nn-1]*tol)) ||
```

```
            (abs(t[nn - 2]-t[j-1])<=abs(t[j-1]*tol))) {
               done = 1;
            } else if (n>15) {
               done = 1;
               error = 1;
            }
         }
      }
      nn = j+1;
   }while(done != 1);
   sumMain = t[j-1];
}
int main() {
   lower = 1.0;
   upper = 9.0;
   printf("\n");
        SAMPLE ;
   romb(lower,upper);
        SAMPLE ;
   printf("\n");
   printf("Area= %f",sumMain);
   return 0;
}
```

ПРИЛОЖЕНИЕ Г.

ФУ ROMB.CPP

```
#include <stdio.h>
#include <math.h>
#include "sampler.h"

double tol = 0.0001;
int done = 0;
double sumMain,upper,lower;

double fx(double x) {
   return 1.0/x;
}

void romb(double lower,double upper){
        SAMPLE ;
   int p;
   int nx[16];
   for (p = 0; p < 16; p++) {
      nx[p] = 0;
   }
   double t[136];
   for (p = 0; p < 136; p++) {
      t[p] = 0.0;
   }
   int done,error;
   int pieces,nt,i,ii,n,nn,l,ntra,k,m,j;
   double delta_x,c,sum,fotom,x;
   pieces = 1;
   nx[0] = 1;
```

```
delta_x = (upper-lower)/pieces;
c = (fx(lower)+fx(upper))*0.5;
t[0] = delta_x*c;
n = 1;
nn = 2;
sum = c;
     SAMPLE ;
do {
   n = n+1;
   fotom = 4.0;
   nx[n-1] =  nn;
   pieces = pieces*2;
   l = pieces-1;
   delta_x = (upper-lower)/pieces;
            SAMPLE ;
   for (ii =1; ii<=((l+1)/2); ii++) {
      i = ii*2-1;
      x = lower+i *delta_x;
      sum = sum+fx(x);
   }
   t[nn-1] = delta_x *sum;
   printf("%d%f ", pieces,t[nn]);
   ntra = nx[n-2];
   k = n-1;
            SAMPLE ;
   for (m = 1;m<=k;m++) {
      j = nn + m;
      nt = nx[n-1-1]+m-1;
      t[j-1] = (fotom *t[j - 2]-t[nt-1])/(fotom-1.0);
   }
```

```
        printf("%d%f\n",j-1,t[j-1]);
                SAMPLE ;
        if (n>4) {
           if (t[nn] < 0.0 || t[nn] > 0.0) {
              if ((abs(t[ntra + 1]-t[nn-1])<=abs(t[nn-1]*tol)) ||
              (abs(t[nn - 2]-t[j-1])<=abs(t[j-1]*tol))) {
                 done = 1;
              } else if (n>15) {
                 done = 1;
                 error = 1;
              }
           }
        }
        nn = j+1;
                SAMPLE ;
    }while(done != 1);
    sumMain = t[j-1];
         SAMPLE ;
}

int main() {
    lower = 1.0;
    upper = 9.0;
    printf("\n");
    romb(lower,upper);
    printf("\n");
    printf("Area= %f",sumMain);
    return 0;
}
```

ПРИЛОЖЕНИЕ Д.
МОДИФИЦИРОВАННАЯ ФУ ROMB.CPP

```
#include <stdio.h>
#include <math.h>
#include "sampler.h"

double tol = 0.0001;
int done = 0;
double sumMain,upper,lower;

double fx(double x) {
   return 1.0/x;
}

void romb(double lower,double upper){
   SAMPLE ;
   int p;
   int nx[16] = {0};
   double t[136]= {0};
   int done,error;
   int pieces,ii,n,nn,l,ntra,k,m,j;
   double delta_x,c,sum,fotom,x;
   pieces = 1;
   nx[0] = 1;
   delta_x = (upper-lower)/pieces;
   c = (fx(lower)+fx(upper))*0.5;
   t[0] = delta_x*c;
   n = 1;
   nn = 2;
   sum = c;
```

```
SAMPLE ;
do {
   n = n+1;
   fotom = 4.0;
   nx[n-1] =  nn;
   pieces = pieces*2;
   l = pieces-1;
   delta_x = (upper-lower)/pieces;
   SAMPLE ;
   for (ii =1; ii<=((l+1)/2); ii++) {
      x = lower+(ii*2-1) *delta_x;
      sum = sum+(1.0/x);
   }
   t[nn-1] = delta_x *sum;
   printf("%d%f ", pieces,t[nn]);
   ntra = nx[n-2];
   k = n-1;
   SAMPLE ;
   for (m = 1;m<=k;m++) {
      j = nn + m;
      t[j-1] = (fotom *t[j - 2]-t[(nx[n-1-1]+m-1)-1])/(fotom-1.0);
   }
   printf("%d%f\n",j-1,t[j-1]);
   SAMPLE ;
   if (n>4) {
      if (t[nn] < 0.0 || t[nn] > 0.0) {
         if ((abs(t[ntra + 1]-t[nn-1])<=abs(t[nn-1]*tol)) ||
         (abs(t[nn - 2]-t[j-1])<=abs(t[j-1]*tol))) {
            done = 1;
         } else if (n>15) {
```

```
                done = 1;
                error = 1;
            }
        }
    }
    nn = j+1;
    SAMPLE ;
  }while(done != 1);
  sumMain = t[j-1];
  SAMPLE ;
}

int main() {
  lower = 1.0;
  upper = 9.0;
  printf("\n");
  romb(lower,upper);
  printf("\n");
  printf("Area= %f",sumMain);
  return 0;
}
```