

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

Кириячиков 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 11.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 11.cpp (исследование каждого цикла в for).

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

Так как в программе присутствует только 1 цикл for, в котором производятся простые операции, то улучшений не предусматривается.

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.

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_CYS.C

```
1  #include <stdlib.h>
2  #include "Sampler.h"
3  #define Size 10000
4  int i, tmp, dim[Size];
5
6  void main()
7  {
8      SAMPLE;
9      for(i=0;i<Size/10;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
10     SAMPLE;
11     for(i=0;i<Size/5;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
12     SAMPLE;
13     for(i=0;i<Size/2;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
14     SAMPLE;
15     for(i=0;i<Size;i++) { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
16     SAMPLE;
17     for(i=0;i<Size/10;i++)
18         { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
19     SAMPLE;
20     for(i=0;i<Size/5;i++)
21         { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
22     SAMPLE;
23     for(i=0;i<Size/2;i++)
24         { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
25     SAMPLE;
26     for(i=0;i<Size;i++)
27         { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
28     SAMPLE;
29     for(i=0;i<Size/10;i++)
30         { tmp=dim[0];
31           dim[0]=dim[i];
32           dim[i]=tmp;
33         };
34     SAMPLE;
35     for(i=0;i<Size/5;i++)
36         { tmp=dim[0];
37           dim[0]=dim[i];
38           dim[i]=tmp;
39         };
40     SAMPLE;
41     for(i=0;i<Size/2;i++)
42         { tmp=dim[0];
43           dim[0]=dim[i];
44           dim[i]=tmp;
45         };
46     SAMPLE;
47     for(i=0;i<Size;i++)
48         { tmp=dim[0];
49           dim[0]=dim[i];
50           dim[i]=tmp;
51         };
52     SAMPLE;
53 }
```

ПРИЛОЖЕНИЕ В

TEST_SUB.C

```
1  #include <stdlib.h>
2  #include "Sample.h"
3  const unsigned Size = 1000;
4
5
6  void TestLoop(int nTimes)
7  {
8      static int TestDim[Size];
9      int tmp;
10     int iLoop;
11
12     while (nTimes > 0)
13     {
14         nTimes --;
15
16         iLoop = Size;
17         while (iLoop > 0)
18         {
19             iLoop -- ;
20             tmp = TestDim[0];
21             TestDim[0] = TestDim[nTimes];
22             TestDim[nTimes] = tmp;
23         }
24     }
25 } /* TestLoop */
26
27
28 void main()
29 {
30     SAMPLE;
31     TestLoop(Size / 10);
32     SAMPLE;
33     TestLoop(Size / 5);
34     SAMPLE;
35     TestLoop(Size / 2);
36     SAMPLE;
37     TestLoop(Size / 1);
38     SAMPLE;
39 }
40
```


ПРИЛОЖЕНИЕ С

Полное время L1FULL.CPP

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "math.h"
4  #include "Sampler.h"
5
6  const double tol= 1.0E-6;
7
8  double fx(double x){
9      return exp(-1.0*x/2.0);
10 }
11
12 double dfx(double x){
13     return (-1.0*exp(-1.0*x/2.0)/2.0);
14 }
15
16 double simps(double lower,double upper,double tol,double sum){
17     double x,delta_x,even_sum,odd_sum,end_sum,end_cor,sum1;
18     int pieces;
19
20     pieces=2;
21     delta_x=(upper-lower)/pieces;
22     odd_sum=fx(lower+delta_x);
23     even_sum=0.0;
24     end_sum=fx(lower)+fx(upper);
25     end_cor=dfx(lower)-dfx(upper);
26     sum=(end_sum+4.0*odd_sum)*delta_x/3.0;
27     do{
28         pieces=pieces*2;
29         sum1=sum;
30         delta_x=(upper-lower)/pieces;
31         even_sum=even_sum+odd_sum;
32         odd_sum=0.0;
33         for (int i=1;i<=pieces/2;i++){
34             x=lower+delta_x*(2.0*i-1.0);
35             odd_sum=odd_sum+fx(x);
36         }
37         sum=(7.0*end_sum+14.0*even_sum+16.00*odd_sum
38             +end_cor*delta_x)*delta_x/15.0;
39     }
40     while ((sum!=sum1) & (abs(sum-sum1)<=abs(tol*sum)));
41     return sum;
42 }
43
44 int main(void) {
45     double sum=0.0;
46     double lower=1.0;
47     double upper=9.0;
48     double res =0.0;
49     SAMPLE;
50     simps(lower,upper,tol,sum);
51     SAMPLE;
52     return 0;
53 }
```

ПРИЛОЖЕНИЕ D

Время ФУ L1.CPP

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "math.h"
4  #include "Sampler.h"
5
6  const double tol= 1.0E-6;
7
8  double fx(double x){
9      return exp(-1.0*x/2.0);
10 }
11
12 double dfx(double x){
13     return (-1.0*exp(-1.0*x/2.0)/2.0);
14 }
15
16 double simps(double lower,double upper,double tol,double sum){
17     double x,delta_x,even_sum,odd_sum,end_sum,end_cor,sum1;
18     int pieces;
19
20     pieces=2;
21     delta_x=(upper-lower)/pieces;
22     odd_sum=fx(lower+delta_x);
23     even_sum=0.0;
24     end_sum=fx(lower)+fx(upper);
25     end_cor=dfx(lower)-dfx(upper);
26     sum=(end_sum+4.0*odd_sum)*delta_x/3.0;
27     SAMPLE;
28     do{
29         pieces=pieces*2;
30         sum1=sum;
31         delta_x=(upper-lower)/pieces;
32         even_sum=even_sum+odd_sum;
33         odd_sum=0.0;
34         SAMPLE;
35         for (int i=1;i<=pieces/2;i++){
36             x=lower+delta_x*(2.0*i-1.0);
37             odd_sum=odd_sum+fx(x);
38         }
39         SAMPLE;
40         sum=(7.0*end_sum+14.0*even_sum+16.00*odd_sum
41             +end_cor*delta_x)*delta_x/15.0;
42     }
43     while ((sum!=sum1) & (abs(sum-sum1)<=abs(tol*sum)));
44     SAMPLE;
45     return sum;
46 }
47
48 int main(void) {
49     double sum=0.0;
50     double lower=1.0;
51     double upper=9.0;
52     double res =0.0;
53     simps(lower,upper,tol,sum);
54     return 0;
55 }
```

ПРИЛОЖЕНИЕ Е

Исследование цикла for L1.CPP

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "math.h"
4  #include "Sampler.h"
5
6  const double tol= 1.0E-6;
7
8  double fx(double x){
9      return exp(-1.0*x/2.0);
10 }
11
12 double dfx(double x){
13     return (-1.0*exp(-1.0*x/2.0)/2.0);
14 }
15
16 double simps(double lower,double upper,double tol,double sum){
17     double x,delta_x,even_sum,odd_sum,end_sum,end_cor,suml;
18     int pieces;
19
20     pieces=2;
21     delta_x=(upper-lower)/pieces;
22     odd_sum=fx(lower+delta_x);
23     even_sum=0.0;
24     end_sum=fx(lower)+fx(upper);
25     end_cor=dfx(lower)-dfx(upper);
26     sum=(end_sum+4.0*odd_sum)*delta_x/3.0;
27     SAMPLE;
28     do{
29         pieces=pieces*2;
30         suml=sum;
31         delta_x=(upper-lower)/pieces;
32         even_sum=even_sum+odd_sum;
33         odd_sum=0.0;
34         SAMPLE;
35         for (int i=1;i<=pieces/2;i++){
36             SAMPLE;
37             x=lower+delta_x*(2.0*i-1.0);
38
39             odd_sum=odd_sum+fx(x);
40         }
41         SAMPLE;
42         sum=(7.0*end_sum+14.0*even_sum+16.00*odd_sum
43             +end_cor*delta_x)*delta_x/15.0;
44     } while ((sum!=suml) & (abs(sum-suml)<=abs(tol*sum)));
45     SAMPLE;
46     return sum;
47 }
48
49 int main(void) {
50     double sum=0.0;
51     double lower=1.0;
52     double upper=9.0;
53     double res =0.0;
54     simps(lower,upper,tol,sum);
55     return 0;
56 }
```