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'а 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 A.

Результаты профилирования:

	NN	 		Имя	обработанного фай	ла	
	1.	TE	ST_CY	C.CPP			
	Ta	блиц	цас р	езульт	атами измерений (используется 13 и:	з 416 записей)
Ис	х.П	юз.	Прием	1.Поз.	Общее время(мкс)	Кол-во прох. Сре,	днее время(мкс)
1	:	8	1:	10	4335.47	1	4335.47
1	:	10	1:	12	8675.98		8675.98
1	:	12	1:	14	21671.50	1	21671.50
1	:			16	43348.87	1	43348.87
1	:		1:	19	4337.15	1	4337.15
1	:	19	1:		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.

Pea	зу.	льта	ать	ЫΠ	рофи	лирования:		
	nn				Имя	обработанного фай	ла	
	1.	TE	ST_	SUE	3.CPP			
	Ta	блиц	ца (с р		атами измерений (
Исх	 .П	03.	Прі	ием	.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				2168480.87		2168480.87
1						4336949.16		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.

P	езульт	гаты 1	профил	іирования	с измерение	ем полно	ого 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 : 39		_	1004.04 42.74	10 1	100.40 42.74

По результатам измерений времени на Φ У 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:

Табли	ща с ј	результ	атами измерений (используется 5	э из 416 записей)
	-		Общее время (мкс)	-	Среднее время (мкс)
1: 27	1:	34	243.89	1	243.89
1: 34			39.39	11	3.58
1: 36	1:	36	245594.13 656.23		60.15 59.66
1: 40			1014.10 43.58	10 1	101.41 43.58

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

Результаты профилирования с измерением времен выполнения измененной программы:

Кол-во прох.	Общее время(мкс)	и.Поз.	1e1	При	Юз.	.П	CX
1	243.89	34	:	1	27	:	1
11	35.20	36	:	1	34	:	1
4083	226545.03	36	:	1	36	:	1
11	601.75	40	:	1	36	:	1
10	1011.58	34	:	1	40	:	1
1	43.58	45		1	40		1
(1 11 4083 11	243.89 1 35.20 11 226545.03 4083 601.75 11	34 243.89 1 36 35.20 11 36 226545.03 4083 40 601.75 11 34 1011.58 10	: 34 243.89 1 : 36 35.20 11 : 36 226545.03 4083 : 40 601.75 11 : 34 1011.58 10	1: 34 243.89 1 1: 36 35.20 11 1: 36 226545.03 4083 1: 40 601.75 11 1: 34 1011.58 10	27 1 : 34 243.89 1 34 1 : 36 35.20 11 36 1 : 36 226545.03 4083 36 1 : 40 601.75 11 40 1 : 34 1011.58 10	: 34 1 : 36

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

```
1 #include <stdlib.h>
     #include "Sampler.h"
 2
 3
     #define Size 10000
 4
     int i, tmp, dim[Size];
 5
 6
     void main()
 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++)
           { tmp=dim[0];
36
37
             dim[0]=dim[i];
38
             dim[i]=tmp;
39
            };
          SAMPLE;
40
41
          for (i=0; i < Size / 2; i++)
42
            { tmp=dim[0];
43
              dim[0]=dim[i];
44
              dim[i]=tmp;
45
            1:
46
          SAMPLE:
47
          for(i=0;i<Size;i++)
48
            { tmp=dim[0];
49
              dim[0]=dim[i];
50
              dim[i]=tmp;
51
            1:
52
          SAMPLE:
53
      }
```

приложение в

TEST_SUB.C

```
1 #include <stdlib.h>
 2 #include "Sample.h"
 3 const unsigned Size = 1000;
 4
 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];
         TestDim[0] = TestDim[nTimes];
21
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>
     #include <stdlib.h>
3
     #include "math.h"
     #include "Sampler.h"
4
 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
          do{
28
             pieces=pieces*2;
29
             suml=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);
                  odd sum=odd_sum+fx(x);
35
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!=suml) & (abs(sum-suml) <= abs(tol*sum)));
41
           return sum;
     L
42
 43
44 Fint main (void) {
45
          double sum=0.0;
46
          double lower=1.0;
47
          double upper=9.0;
48
          double res =0.0;
 49
          SAMPLE;
           simps(lower,upper,tol,sum);
 50
          SAMPLE:
 51
 52
          return 0;
     L}
 53
```

приложение р

Время ФУ L1.СРР

```
1 #include <stdio.h>
    #include <stdlib.h>
 3
    #include "math.h"
 4
    #include "Sampler.h"
 5
 6
    const double tol= 1.0E-6;
 7
    ☐double fx(double x){
          return exp(-1.0*x/2.0);
10
11
    ⊟double dfx(double x) {
12
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{
              pieces=pieces*2;
29
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++){</pre>
    36
                 x=lower+delta_x*(2.0*i-1.0);
37
                  odd sum=odd sum+fx(x);
38
39
              SAMPLE;
    \phi
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!=suml) & (abs(sum-suml) <= 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
     L
```

приложение е

Исследование цикла for L1.CPP

```
#include <stdio.h>
1
 2
      #include <stdlib.h>
      #include "math.h"
 3
 4
      #include "Sampler.h"
 5
     const double tol= 1.0E-6;
 6
 7
 8
    ∃double fx(double x){
          return exp(-1.0*x/2.0);
 9
10
11
12
    ∃double dfx(double x){
13
          return (-1.0*exp(-1.0*x/2.0)/2.0);
14
15
    double simps(double lower, double upper, double tol, double sum) {
16
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;
      end sum=fx(lower)+fx(upper);
24
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;
              odd_sum=0.0;
33
34
               SAMPLE;
               for (int i=1;i<=pieces/2;i++) {</pre>
35
36
                  SAMPLE;
37
                  x=lower+delta_x*(2.0*i-1.0);
38
                  odd_sum=odd_sum+fx(x);
39
40
              SAMPLE;
    ŧ
41
              sum=(7.0*end_sum+14.0*even_sum+16.00*odd_sum
42
                      +end_cor*delta_x)*delta_x/15.0;
43
44
          while ((sum!=suml) & (abs(sum-suml)<=abs(tol*sum)));</pre>
          SAMPLE:
45
          return sum;
46
     L
47
48
49
    ☐int main(void) {
50
         double sum=0.0;
51
          double lower=1.0;
52
          double upper=9.0;
          double res =0.0;
53
54
          simps(lower,upper,tol,sum);
55
          return 0:
56
```

приложение г

Измененная программа L1.СРР

```
1 #include <stdio.h>
     #include <stdlib.h>
     #include "math.h"
 3
     #include "Sampler.h"
 4
 6
     const double tol= 1.0E-6;
 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
    Hodouble 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++){</pre>
             SAMPLE;
36
37
                x=lower+delta x*(2.0*i-1.0);
                 odd sum=odd sum+exp(-1.0*x/2.0);
38
39
40
             SAMPLE;
41
              sum=(7.0*end sum+14.0*even sum+16.00*odd sum
                     +end_cor*delta_x) *delta_x/15.0;
42
43
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;
          double lower=1.0;
51
52
         double upper=9.0;
53
         double res =0.0;
54
          simps(lower,upper,tol,sum);
55
          return 0;
56
```