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_v2»

Студент гр. 8304 _____ Мухин А. М.

Преподаватель _____ Кирьянчиков В. А.

Санкт-Петербург
2022

Цель работы.

Изучить возможности измерения динамических характеристик программ с помощью профилировщиков на примере профилировщика SAMPLER_v2.

Задание.

1) Выполнить под управлением SAMPLER тестовые программы test_cyc.c и test_sub.c и привести отчет по результатам их выполнения с анализом параметров повторения циклов, структуры описания циклов, способов профилирования процедур и проверкой их влияния на точность и чувствительность профилирования.
2) Разработанную в лаб. работе 1 программу, реализующую заданный вычислительный алгоритм, разбить на функциональные участки (ФУ) и расставить на их границах контрольные точки (КТ) для выполнения с помощью ПИМ SAMPLER измерений и получения профиля выполнения программы, представляющего времена выполнения и количество выполнений каждого ФУ.
3) Скомпилировать полученную программу. При компиляции добавить путь к sampler.h в набор путей поиска включаемых файлов (`Isampler/libsampler` при компиляции, если архив был распакован в текущий каталог), при линковке добавить путь к libsampler.a в набор путей поиска библиотек и подключить её (флаги `-LSampler/build/libsampler -lsampler` при линковке).
4) Выполнить скомпилированную программу под управлением Sampler'а с внешним зацикливанием и получить отчет по результатам профилирования. Зацикливание можно выполнять при помощи программы sampler-repeat. Использование программы приведено в разделе 4 документа «Описание работы с ПИМ SAMPLER_v2». Число повторов зависит от сложности самой программы; имеет смысл начальное число запусков взять равным 10 и увеличивать его в 5–10 раз до тех пор, пока среднее время выполнения участков не стабилизируется, или на запуски станет уходить слишком много

времени, или на результаты станет уходить слишком много дискового пространства.

5) Проанализировать полученный отчет и выявить "узкие места", приводящие к ухудшению производительности программы.
6) Ввести в программу усовершенствования для повышения производительности, получить новые профили, добавить их в отчет и объяснить полученные результаты.

Ход работы.

Были выполнены под управлением монитора SAMPLER тестовые программы test_cyc.c и test_sub.c. Результаты представлены на рисунках 1 и 2.

исх	прием	общее время	кол-во проходов	среднее время
13	15	2927.987	1	2927.987
15	17	5970.227	1	5970.227
17	19	23430.207	1	23430.207
19	21	31067.927	1	31067.927
21	24	2791.465	1	2791.465
24	27	7458.920	1	7458.920
27	30	18776.657	1	18776.657
30	33	33274.726	1	33274.726
33	39	2751.528	1	2751.528
39	45	6774.269	1	6774.269
45	51	17409.092	1	17409.092
51	57	35964.209	1	35964.209

Рисунок 1 – Результат работы SAMPLER_v2 для программы test_cyc.c

исх	прием	общее время	кол-во проходов	среднее время
30	32	29300426.089	1	29300426.089
32	34	58189365.378	1	58189365.378
34	36	145513887.433	1	145513887.433
36	38	290886160.167	1	290886160.167

Рисунок 2 - Результат работы SAMPLER_v2 для программы test_sub.c

Под управлением монитора SAMPLER_v2 была выполнена немного измененная программа из лабораторной работы №1. Результат измерений для полного времени выполнения функции *solve* представлен на рисунке 3. Исходный код этой программы представлен в Приложении А.

исх	прием	общее время	кол-во проходов	среднее время
75	77	355.713	1	355.713

Рисунок 3 - Результат работы SAMPLER_v2 для измерения полного времени выполнения функции

Было выполнено разбиение данной программы на функциональные участки. Исходный код программы, разбитый на функциональные участки, представлен в приложении Б. Полученные с помощью программы SAMPLER_v2 результаты представлены на рисунке 4.

исх	прием	общее время	кол-во проходов	среднее время
95	54	26.678	1	26.678
54	58	12.920	1	12.920
58	59	5.492	1	5.492
59	60	11.291	3	3.764
59	68	6.143	1	6.143
60	61	1.082	3	0.361
61	62	48.202	9	5.356
61	66	21.433	3	7.144
62	64	25.298	9	2.811
64	61	63.935	9	7.104
66	59	5.431	3	1.810
68	71	34.049	1	34.049
71	77	9.672	1	9.672
77	33	14.071	1	14.071
33	35	4.889	3	1.630
35	36	15.964	3	5.321
36	37	71.668	9	7.963
36	47	26.473	3	8.824
37	39	56.011	9	6.223
39	45	10.579	3	3.526
39	41	15.638	6	2.606
45	36	12.152	9	1.350
47	49	69.021	3	23.007
49	33	36.090	2	18.045
49	81	20.337	1	20.337
41	43	50.822	6	8.470
43	45	21.550	6	3.592
81	83	1.874	1	1.874
83	97	20.419	1	20.419

Рисунок 4 - Результат работы SAMPLER_v2 для измерения полного времени выполнения функции, разбитой на функциональные участки

В итоге общее время выполнения – 719,184 мкс. Разницу в 363,713 мкс с измерением для полного времени можно объяснить вызовом функции измерения.

Как видно из результатов измерения времени выполнения функциональных участков – наиболее затратными фрагментами являются update-операция цикла FOR (строчки 64-61), проверка условия цикла FOR (строчки 36-37), вычисление коэффициентов (строчки 47-49) и присваивание одного элемента массива другому (строчки 37-39).

Для решения этой проблемы в строчках 61, 59 и 36 оператор инкремента был заменен на префиксный, в строчке 48, а также в строчке 38 операция доступа к элементу массива по индексу была заменена адресной арифметикой.

Была выполнена проверка изменённой программы. Результат представлен на рисунке 5. Исходный код модифицированной программы представлен в Приложении В.

исх	прием	общее время	кол-во проходов	среднее время
95	54	25.272	1	25.272
54	58	12.653	1	12.653
58	59	7.454	1	7.454
59	60	14.897	3	4.966
59	68	6.354	1	6.354
60	61	3.447	3	1.149
61	62	54.430	9	6.048
61	66	21.868	3	7.289
62	64	29.979	9	3.331
64	61	3.005	9	0.334
66	59	3.538	3	1.179
68	71	28.293	1	28.293
71	77	13.530	1	13.530
77	33	13.305	1	13.305
33	35	2.868	3	0.956
35	36	13.663	3	4.554
36	37	50.516	9	5.613
36	47	24.954	3	8.318
37	39	44.962	9	4.996
39	45	7.337	3	2.446
39	41	14.228	6	2.371
45	36	6.157	9	0.684
47	49	70.547	3	23.516
49	33	36.895	2	18.448
49	81	13.795	1	13.795
41	43	44.079	6	7.347
43	45	9.763	6	1.627
81	83	1.267	1	1.267
83	97	22.305	1	22.305

Рисунок 5 - Результат работы SAMPLER_v2 для измерения полного времени выполнения оптимизированной функции, разбитой на функциональные участки

В результате внесённых изменений общее время выполнения – 601,22 мкс. Удалось добиться снижения времени выполнения на 117,964 мкс (16 %).

Выводы.

В ходе выполнения лабораторной работы были изучены возможности измерения динамических характеристик программ с помощью профилировщиков на примере профилировщика SAMPLER_v2. Для программы, взятой из первой лабораторной работы, было выполнено измерение времени работы, с последующим выявлением узких мест и их устранения. В результате чего удалось добиться более быстрого выполнения программы.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "sampler.h"
4
5 #define RMAX 3
6 #define CMAX 3
7
8
9 void get_data(double a[RMAX][CMAX], double y[CMAX]) {
10  a[0][0] = 1;
11  a[0][1] = -43;
12  a[0][2] = 19;
13  y[0] = 81;
14  a[1][0] = 145;
15  a[1][1] = -134;
16  a[1][2] = 99;
17  y[1] = 12;
18  a[2][0] = 325;
19  a[2][1] = 991;
20  a[2][2] = -199;
21  y[2] = 213;
22 }
23
24
25 double deter(double a[RMAX][CMAX]) {
26  return(a[0][0] * (a[1][1] * a[2][2] - a[2][1] * a[1][2])
27       - a[0][1] * (a[1][0] * a[2][2] - a[2][0] * a[1][2])
28       + a[0][2] * (a[1][0] * a[2][1] - a[2][0] * a[1][1]));
29 }
30
31
32 void setup(double a[RMAX][CMAX], double b[RMAX][CMAX], double
y[CMAX], double coef[CMAX], int j, double det) {
33  int i;
34  for (i = 0; i < RMAX; i++) {
35       b[i][j] = y[i];
36       if (j > 0) {
37            b[i][j - 1] = a[i][j - 1];
38       }
39  }
40  coef[j] = deter(b) / det;
41 }
42
43
44  void  solve(double  a[RMAX][CMAX],  double  y[CMAX],  double
coef[CMAX]) {
45  double b[RMAX][CMAX];
46  int i, j;
47  double det;
48
49  for (i = 0; i < RMAX; i++) {
```

```
50        for (j = 0; j < CMAX; j++) {
51             b[i][j] = a[i][j];
52        }
53   }
54
55   det = deter(b);
56   if (det == 0) {
57        return;
58   }
59   else {
60        setup(a, b, y, coef, 0, det);
61        setup(a, b, y, coef, 1, det);
62        setup(a, b, y, coef, 2, det);
63   }
64  }
65
66
67  int main(int argc, char **argv)
68  {
69        sampler_init(&argc, argv);
70   double a[RMAX][CMAX];
71   double y[CMAX];
72   double coef[CMAX];
73
74   get_data(a, y);
75   SAMPLE;
76   solve(a, y, coef);
77   SAMPLE;
78
79   return 0;
80  }
```

ПРИЛОЖЕНИЕ Б.

КОД ПРОГРАММЫ С РАЗДЕЛЕНИЕМ НА ФУ

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include "sampler.h"
 4
 5 #define RMAX 3
 6 #define CMAX 3
 7
 8
 9 void get_data(double a[RMAX][CMAX], double y[CMAX]) {
10  a[0][0] = 1;
11  a[0][1] = -43;
12  a[0][2] = 19;
13  y[0] = 81;
14  a[1][0] = 145;
15  a[1][1] = -134;
16  a[1][2] = 99;
17  y[1] = 12;
18  a[2][0] = 325;
19  a[2][1] = 991;
20  a[2][2] = -199;
21  y[2] = 213;
22 }
23
24
25 double deter(double a[RMAX][CMAX]) {
26  return(a[0][0] * (a[1][1] * a[2][2] - a[2][1] * a[1][2])
27       - a[0][1] * (a[1][0] * a[2][2] - a[2][0] * a[1][2])
28       + a[0][2] * (a[1][0] * a[2][1] - a[2][0] * a[1][1]));
29 }
30
31
32 void setup(double a[RMAX][CMAX], double b[RMAX][CMAX], double
y[CMAX], double coef[CMAX], int j, double det) {
33  SAMPLE;
34  int i;
35  SAMPLE;
36  for (i = 0; SAMPLE, i < RMAX; i++) {
37       SAMPLE;
38       b[i][j] = y[i];
39       SAMPLE;
40       if (j > 0) {
41            SAMPLE;
42            b[i][j - 1] = a[i][j - 1];
43            SAMPLE;
44       }
45       SAMPLE;
46  }
47  SAMPLE;
48  coef[j] = deter(b) / det;
```

```
 49  SAMPLE;
 50 }
 51
 52
 53  void  solve(double  a[RMAX][CMAX],  double  y[CMAX],  double
coef[CMAX]) {
 54  SAMPLE;
 55  double b[RMAX][CMAX];
 56  int i, j;
 57  double det;
 58  SAMPLE;
 59  for (i = 0; SAMPLE, i < RMAX; i++) {
 60       SAMPLE;
 61       for (j = 0; SAMPLE, j < CMAX; j++) {
 62            SAMPLE;
 63            b[i][j] = a[i][j];
 64            SAMPLE;
 65       }
 66       SAMPLE;
 67  }
 68  SAMPLE;
 69
 70  det = deter(b);
 71  SAMPLE;
 72  if (det == 0) {
 73       SAMPLE;
 74       return;
 75  }
 76  else {
 77       SAMPLE;
 78       setup(a, b, y, coef, 0, det);
 79       setup(a, b, y, coef, 1, det);
 80       setup(a, b, y, coef, 2, det);
 81       SAMPLE;
 82  }
 83  SAMPLE;
 84 }
 85
 86
 87 int main(int argc, char **argv)
 88 {
 89       sampler_init(&argc, argv);
 90  double a[RMAX][CMAX];
 91  double y[CMAX];
 92  double coef[CMAX];
 93
 94  get_data(a, y);
 95  SAMPLE;
 96  solve(a, y, coef);
 97  SAMPLE;
 98
 99  return 0;
100 }
```

ПРИЛОЖЕНИЕ В.

КОД ПРОГРАММЫ С ОПТИМИЗАЦИЯМИ

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "sampler.h"
4
5 #define RMAX 3
6 #define CMAX 3
7
8
9 void get_data(double a[RMAX][CMAX], double y[CMAX]) {
10  a[0][0] = 1;
11  a[0][1] = -43;
12  a[0][2] = 19;
13  y[0] = 81;
14  a[1][0] = 145;
15  a[1][1] = -134;
16  a[1][2] = 99;
17  y[1] = 12;
18  a[2][0] = 325;
19  a[2][1] = 991;
20  a[2][2] = -199;
21  y[2] = 213;
22 }
23
24
25 double deter(double a[RMAX][CMAX]) {
26  return(a[0][0] * (a[1][1] * a[2][2] - a[2][1] * a[1][2])
27       - a[0][1] * (a[1][0] * a[2][2] - a[2][0] * a[1][2])
28       + a[0][2] * (a[1][0] * a[2][1] - a[2][0] * a[1][1]));
29 }
30
31
32 void setup(double a[RMAX][CMAX], double b[RMAX][CMAX], double
y[CMAX], double coef[CMAX], int j, double det) {
33  SAMPLE;
34  int i;
35  SAMPLE;
36  for (i = 0; SAMPLE, i < RMAX; ++i) {
37       SAMPLE;
38       b[i][j] = *(y + i);
39       SAMPLE;
40       if (j > 0) {
41            SAMPLE;
42            b[i][j - 1] = a[i][j - 1];
43            SAMPLE;
44       }
45       SAMPLE;
46  }
47  SAMPLE;
48  *(coef +j) = deter(b) / det;
```

```
 49  SAMPLE;
 50 }
 51
 52
 53  void  solve(double  a[RMAX][CMAX],  double  y[CMAX],  double
coef[CMAX]) {
 54  SAMPLE;
 55  double b[RMAX][CMAX];
 56  int i, j;
 57  double det;
 58  SAMPLE;
 59  for (i = 0; SAMPLE, i < RMAX; ++i) {
 60       SAMPLE;
 61       for (j = 0; SAMPLE, j < CMAX; ++j) {
 62            SAMPLE;
 63            b[i][j] = a[i][j];
 64            SAMPLE;
 65       }
 66       SAMPLE;
 67  }
 68  SAMPLE;
 69
 70  det = deter(b);
 71  SAMPLE;
 72  if (det == 0) {
 73       SAMPLE;
 74       return;
 75  }
 76  else {
 77       SAMPLE;
 78       setup(a, b, y, coef, 0, det);
 79       setup(a, b, y, coef, 1, det);
 80       setup(a, b, y, coef, 2, det);
 81       SAMPLE;
 82  }
 83  SAMPLE;
 84 }
 85
 86
 87 int main(int argc, char **argv)
 88 {
 89       sampler_init(&argc, argv);
 90  double a[RMAX][CMAX];
 91  double y[CMAX];
 92  double coef[CMAX];
 93
 94  get_data(a, y);
 95  SAMPLE;
 96  solve(a, y, coef);
 97  SAMPLE;
 98
 99  return 0;
100 }
```