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. 6304 _____ Пискунов Я.А.

Преподаватель _____ Кирьянчиков В.А.

Санкт-Петербург
2020

Цель работы.

Изучение метода измерения динамической сложности программ с помощью SAMPLER.

Формулировка задания

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. Замечание: Следует внимательно подойти к выбору ФУ для получения хороших результатов профилирования.

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

Примечания

1. Для трансляции программ следует использовать компиляторы Turbo C++,

ver.3.0 (3.1, 3.2).

2. Для выполнения работы лучше использовать более новую версию монитора Sampler_new и выполнять ее на 32-разрядной машине под управлением ОС не выше Widows XP (при отсутствии реальных средств можно использовать виртуальные).

В этом случае результаты профилирования будут близки к реальным.

Если использовать более старую версию монитора Sampler_old, то ее следует запускать под эмулятором DOSBox-0.74. При этом времена, полученные в профилях будут сильно (примерно в 10 раз) завышены из-за накладных затрат эмулятора, но относительные соотношения временных затрат будут корректны.

3. Если автоматически подобрать время коррекции правильно не удается (это видно по большим значениям измерений времени для коротких фраг1

ментов программы, если время коррекции недостаточно, либо по большому количеству нулевых отсчетов, если время коррекции слишком велико), то следует подобрать подходящее время коррекции ручным способом, уменьшая или увеличивая его в нужную сторону.

4. Так как чувствительность SAMPLER’а по времени достаточно высока (на уровне единиц микросекунд), то вводить вспомогательное зацикливание программы обычно не требуется. Но если измеренные времена явно некорректны, следует ввести зацикливание выполнения программы в 10–100 раз. При этом для каждого повторения выполнения программы следует использовать одни и те же исходные данные.

5. Для обеспечения проверки представляемых вами профилей преподавателем необходимо выполнить нумерацию строк кода программы, соответствующую нумерации строк, указанной в профиле. У преподавателя нет времени для подсчета номеров строк в отчете каждого студента.

Ход работы

Работа выполнялась с использованием старого SAMPLER’a. Так как из четырех различных версий компилятора turbo c++ ни одна не смогла быть установлена на виртуальную машину, то для выполнения работы использовался компилятор Borland C++. В качестве операционной системы, на которой производится компиляция, линковка и применение SAMPLER’a использовался виртуальный образ Windows XP Professional SP3 x32.

В тестовые программы подключен файл Sampler.h и расставлены точки измерения. Измененные программы с пронумерованными строками представлены в приложении А.

Результаты профилирования программ TEST_CYC и TEST_SUB представлены на рис. 1 и рис. 2 соответственно.

```
---------------------------------------------------------------------------
  1.   ..\..\LAB3\TEST\TEST_C.CPP
---------------------------------------------------------------------------


  ’ ў«Ёж  б аЃ§г«мв в ¬Ё Ё§¬ЃаЃ-Ё© ( ЁбЇ®«м§гЃвбп 13 Ё§ 416 § ЇЁбЃ© )

---------------------------------------------------------------------------
€бе.Ц®§. ЦаЁЃ¬.Ц®§.  ЋўЙЃЃ ўаЃ¬п(¬Єб)  Љ®«-ў® Їа®е. ‘аЃ¤-ЃЃ ўаЃ¬п(¬Єб)
---------------------------------------------------------------------------
 1 :   7  1 :   9                1.68            1                1.68
---------------------------------------------------------------------------
 1 :   9  1 :  11                2.51            1                2.51
---------------------------------------------------------------------------
 1 :  11  1 :  13                6.70            1                6.70
---------------------------------------------------------------------------
 1 :  13  1 :  15               13.41            1               13.41
---------------------------------------------------------------------------
 1 :  15  1 :  18                1.68            1                1.68
---------------------------------------------------------------------------
 1 :  18  1 :  21                2.51            1                2.51
---------------------------------------------------------------------------
 1 :  21  1 :  24                6.70            1                6.70
---------------------------------------------------------------------------
 1 :  24  1 :  27               13.41            1               13.41
---------------------------------------------------------------------------
 1 :  27  1 :  33                0.84            1                0.84
---------------------------------------------------------------------------
 1 :  33  1 :  39                1.68            1                1.68
---------------------------------------------------------------------------
 1 :  39  1 :  45                6.70            1                6.70
---------------------------------------------------------------------------
 1 :  45  1 :  51               12.57            1               12.57
---------------------------------------------------------------------------
```

Рисунок 1 – результат профилирования программы TEST_CYC

```
-------------------------------------------------------------------------
  NN           €¬п ®Ўа Ў®в --®J® д ©«
-------------------------------------------------------------------------
  1.   ..\..\LAB3\TEST\TEST_S.CPP
-------------------------------------------------------------------------


  ’ Ў«Ёж  6 аЃ§г«мв в ¬Ё Ё§¬ЃаЃ-Ё© ( ЁбЇ®«м§гЃвбп 5 Ё§ 416 § ЇЁбЃ© )

-------------------------------------------------------------------------
€бе.Ц®§. ЦаЁЃ¬.Ц®§.  ЋЎйЃЃ ўаЃ¬п(¬Єб)  Љ®«-ў® Їа®е. ‘аЃ¤-ЃЃ ўаЃ¬п(¬Єб)
-------------------------------------------------------------------------
 1 :  29  1 :  31             121.52            1             121.52
-------------------------------------------------------------------------
 1 :  31  1 :  33             215.39            1             215.39
-------------------------------------------------------------------------
 1 :  33  1 :  35             541.41            1             541.41
-------------------------------------------------------------------------
 1 :  35  1 :  37            1097.91            1            1097.91
-------------------------------------------------------------------------
```

Рисунок 2 – результат профилирования программы TEST_SUB

Проанализоровав результаты, можно отметить, что на время выполнения циклов влияет только число итераций в них. При этом, зависимость можно считать линейной.

Далее произведем аналогичные действия с программой из лабораторной работы 1. Для удобства работы программа была видоизменена. Так, основная функция программы выполняется один раз без взаимодействия с пользователем. Также подключен заголовочный файл и расставлены точки измерений. В приложении Б представлен код программы с точками измерения полного времени, а в приложении В – с точками для измерения времени выполнения ФУ. Как видно, точки расставлены с целью профилирования основной процедуры программы – solve.

Результаты профилирования представлены соответственно на рис. 3 и рис. 4. Как можно заметить, время выполнения в этих случаях различается более чем в два раза. Причина такого поведения непонятна. Самое большое время здесь у операции между строками 67 и 69 – обычного присваивания. Возможно, дело в использовании эмулятора. Если принять, что время выполнения этой операции 0.84, то общее время становится более похожим на тот результат, который получен в измерении общего времени.

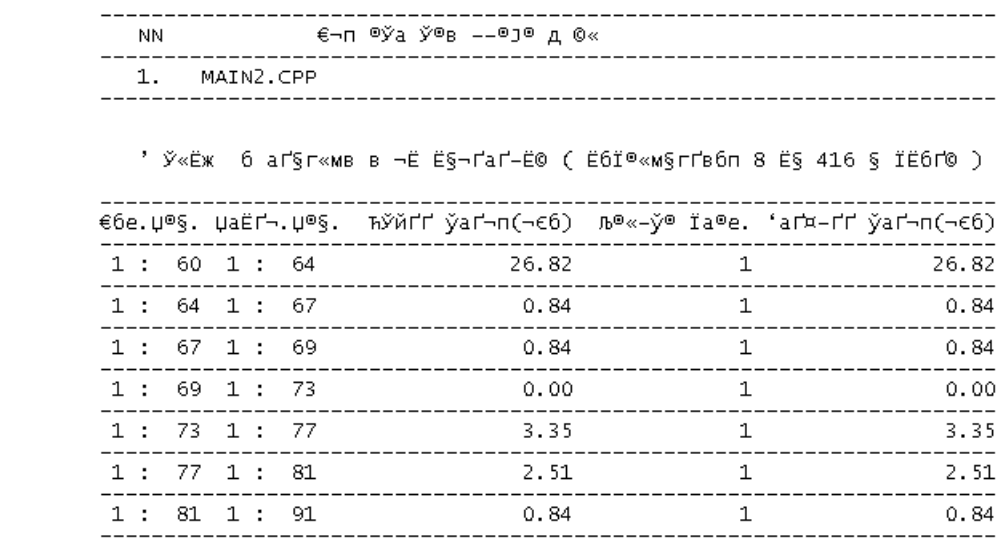
```
-------------------------------------------------------------------------
   NN            €¬п ®Ўа Ў®в --®J® д ©«
-------------------------------------------------------------------------
   1.   MAIN.CPP
-------------------------------------------------------------------------

   ’ Ў«Ёж  6 аЃ§г«мв в ¬Ё Ё§¬ЃаЃ-Ё© ( ЁбЇ®«м§гЃвбп 2 Ё§ 416 § ЇЁбЃ© )

-------------------------------------------------------------------------
€бе.Ц®§. ЦаЁЃ¬.Ц®§.  ЋЎйЃЃ ўаЃ¬п(¬Єб)  Љ®«-ў® Їа®е. ‘аЃ¤-ЃЃ ўаЃ¬п(¬Єб)
-------------------------------------------------------------------------
 1 :  94  1 :  96                6.70             1                6.70
-------------------------------------------------------------------------
```

Рисунок 3 – результат профилирования ЛР1 (полное время)

```
-------------------------------------------------------------------------
   NN            €¬п ®Ўа Ў®в --®J® д ©«
-------------------------------------------------------------------------
   1.   MAIN.CPP
-------------------------------------------------------------------------

   ’ Ў«Ёж  6 аЃ§г«мв в ¬Ё Ё§¬ЃаЃ-Ё© ( ЁбЇ®«м§гЃвбп 8 Ё§ 416 § ЇЁбЃ© )

-------------------------------------------------------------------------
€бе.Ц®§. ЦаЁЃ¬.Ц®§.  ЋЎйЃЃ ўаЃ¬п(¬Єб)  Љ®«-ў® Їа®е. ‘аЃ¤-ЃЃ ўаЃ¬п(¬Єб)
-------------------------------------------------------------------------
 1 :  60  1 :  64                0.84             1                0.84
-------------------------------------------------------------------------
 1 :  64  1 :  67                0.84             1                0.84
-------------------------------------------------------------------------
 1 :  67  1 :  69                8.38             1                8.38
-------------------------------------------------------------------------
 1 :  69  1 :  73                0.84             1                0.84
-------------------------------------------------------------------------
 1 :  73  1 :  75                3.35             1                3.35
-------------------------------------------------------------------------
 1 :  75  1 :  79                2.51             1                2.51
-------------------------------------------------------------------------
 1 :  79  1 :  89                0.84             1                0.84
-------------------------------------------------------------------------
```

Рисунок 4 - результат профилирования ЛР1 (время ФУ)

Как можно отметить по результатам профилировки ФУ, наибольшее время занимает непосредственно вычисление (если не считать операции присваисвания). Можно попытаться сократить расходы путем замены вызова функции на непосредственно выполение опреаций. Результат профилирования обновленной программы представлен на рис. 5.



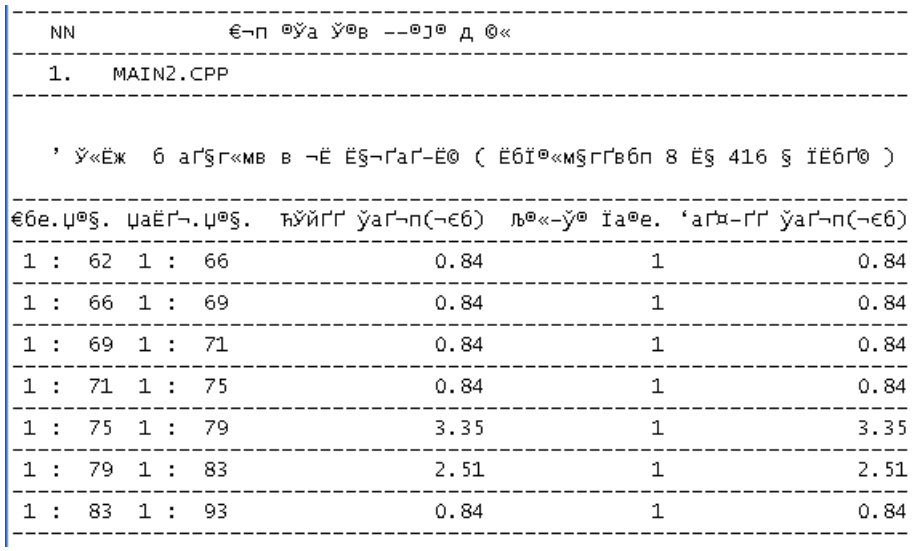
```
-----------------------------------------------------------------------
   NN            €¬п ®Ўа Ў®в --®J® д ©«
-----------------------------------------------------------------------
   1.   MAIN2.CPP
-----------------------------------------------------------------------

   ' Ў«Ёж  б аЃ§г«мв в ¬Ё Ё§¬ЃаЃ-Ё© ( ЁбЇ®«м§гЃвбп 8 Ё§ 416 § ЇЁбЃ® )

-----------------------------------------------------------------------
€бе.Ц®§. ЦаЁЃ¬.Ц®§.  ЋЎЙЃЃ ЎаЃ¬п(¬€б)  Љ®«-Ў® Їа®е. ‘аЃ¤-ЃЃ ЎаЃ¬п(¬€б)
-----------------------------------------------------------------------
 1 :  60  1 :  64               26.82            1               26.82
-----------------------------------------------------------------------
 1 :  64  1 :  67                0.84            1                0.84
-----------------------------------------------------------------------
 1 :  67  1 :  69                0.84            1                0.84
-----------------------------------------------------------------------
 1 :  69  1 :  73                0.00            1                0.00
-----------------------------------------------------------------------
 1 :  73  1 :  77                3.35            1                3.35
-----------------------------------------------------------------------
 1 :  77  1 :  81                2.51            1                2.51
-----------------------------------------------------------------------
 1 :  81  1 :  91                0.84            1                0.84
-----------------------------------------------------------------------
```

Рисунок 5 – результат попытки оптимизации программы

Как можно заметить, появилось новое аномальное время между строками 60 и 64. А вот предыдущее теперь и вовсе имеет нулевое значение. Догадка о влиянии виртуальной машины оказалась верна, однако цели уменьшения времени выполнения добиться не удалось. Третьей попыткой была замена вызова фнукции deter в setup на ее тело. Измененная функция представлена на рис. 6, а результат профилирования на рис. 7. Точки измерения не изменились не считая общего сдвига значений номеров строк из-за двух описанных измерений в общем на четыре.

```
void setup(double** b, double* coef, int j){
    int i;
    for(i=0; i<n; i++){
        b[i][j] = y[i];
        if (j>0)
            b[i][j-1] = a[i][j-1];
    }
    coef[j] = (b[0][0]*(b[1][1]*b[2][2]-b[2][1]*b[1][2])
               -b[0][1]*(a[1][0]*b[2][2]-b[2][0]*b[1][2])
               +b[0][2]*(b[1][0]*b[2][1]-b[2][0]*b[1][1])) / det;
}
```

Рисунок 6 – попытка изменения программы



```
-----------------------------------------------------------------------
  NN            €¬п ®Ўа Ў®в --®j® д ©«
-----------------------------------------------------------------------
  1.   MAIN2.CPP
-----------------------------------------------------------------------

  ' Ў«Ёж  б аЃ§г«мв в ¬Ё Ё§¬ЃаЃ-Ё© ( ЁбЇ®«м§гЃвбп 8 Ё§ 416 § ЇЁбЃ© )

-----------------------------------------------------------------------
€бе.Џ®§. Џа ЁЃ¬.Џ®§.  ЋЎЙЃЃ ЎаЃ¬п(¬€б)  Љ®«-Ў® Їа®е. ‘аЃ¤-ЃЃ ЎаЃ¬п(¬€б)
-----------------------------------------------------------------------
 1 :  62  1 :  66                0.84             1                0.84
-----------------------------------------------------------------------
 1 :  66  1 :  69                0.84             1                0.84
-----------------------------------------------------------------------
 1 :  69  1 :  71                0.84             1                0.84
-----------------------------------------------------------------------
 1 :  71  1 :  75                0.84             1                0.84
-----------------------------------------------------------------------
 1 :  75  1 :  79                3.35             1                3.35
-----------------------------------------------------------------------
 1 :  79  1 :  83                2.51             1                2.51
-----------------------------------------------------------------------
 1 :  83  1 :  93                0.84             1                0.84
-----------------------------------------------------------------------
```

Рисунок 6 – результат второй попытки оптимизации

Вероятно, улучшения времени выполнения не происходит из-за использования глобальных переменных. Также в целом время выполнения достаточно маленькое и результат оптимизации может быть незаметен.

Выводы.

В ходе выполнения данной работы был изучен профилировщик SAMPLER, на практике проведены вычисления профилей программ. Также было проанализировано полное время работы программы и время выполнения ФУ. Установлено, что использование виртуальной машины оказывает влияние на корректность получаемых результатов. Произведена попытка частичной оптимизации, которая, однако, завершилась неудачей.

ПРИЛОЖЕНИЕ А

ИЗМЕНЕННЫЕ ТЕСТОВЫЕ ПРОГРАММЫ

TEST_CYC.CPP

```
1 #include "Sampler.h"
2 #define Size 10000
3 int i, tmp, dim[Size];
4
5 void main()
6 {
7     SAMPLE;
8     for(i=0;i<Size/10;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
9     SAMPLE;
10    for(i=0;i<Size/5;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
11    SAMPLE;
12    for(i=0;i<Size/2;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
13    SAMPLE;
14    for(i=0;i<Size;i++)  { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
15    SAMPLE;
16    for(i=0;i<Size/10;i++)
17      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
18    SAMPLE;
19    for(i=0;i<Size/5;i++)
20      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
21    SAMPLE;
22    for(i=0;i<Size/2;i++)
23      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
24    SAMPLE;
25    for(i=0;i<Size;i++)
26      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
27    SAMPLE;
28    for(i=0;i<Size/10;i++)
29      { tmp=dim[0];
30        dim[0]=dim[i];
31        dim[i]=tmp;
32      };
33    SAMPLE;
34    for(i=0;i<Size/5;i++)
35      { tmp=dim[0];
36        dim[0]=dim[i];
37        dim[i]=tmp;
```

```
38      };
39    SAMPLE;
40    for(i=0;i<Size/2;i++)
41      { tmp=dim[0];
42        dim[0]=dim[i];
43        dim[i]=tmp;
44      };
45    SAMPLE;
46    for(i=0;i<Size;i++)
47      { tmp=dim[0];
48        dim[0]=dim[i];
49        dim[i]=tmp;
50      };
51    SAMPLE;
52 }
```

TEST_SUB.CPP

```
1  const unsigned int Size = 1000;
2
3
4  void TestLoop(int nTimes)
5  {
6   static int TestDim[Size];
7   int tmp;
8   int iLoop;
9
10  while (nTimes > 0)
11  {
12    nTimes --;
13
14    iLoop = Size;
15    while (iLoop > 0)
16    {
17      iLoop -- ;
18      tmp = TestDim[0];
19      TestDim[0] = TestDim[nTimes];
20      TestDim[nTimes] = tmp;
21    }
22  }
23 } /* TestLoop */
```

```
24
25
26  void main()
27 {
28   TestLoop(Size / 10); // 100 * 1000  Ї®ўв®аЃ-Ё©
29   TestLoop(Size / 5);  // 200 * 1000  Ї®ўв®аЃ-Ё©
30   TestLoop(Size / 2);  // 500 * 1000  Ї®ўв®аЃ-Ё©
31   TestLoop(Size / 1);  // 1000* 1000  Ї®ўв®аЃ-Ё©
32 }
```

ПРИЛОЖЕНИЕ Б

ИЗМЕНЕННЫЙ КОД ЛР1 ДЛЯ ПОЛНОГО ВРЕМЕНИ

```
1  #include "Sampler.sh"
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define RMAX 3
6  #define CMAX 3
7
8  int n;
9  char yesno;
10 int oshibka;
11
12 double* y;
13 double* coef;
14 double** a;
15
16 double det;
17
18 void get_data(double** a, double* y){
19     a[0][0] = 1;
20     a[0][1] = -43;
21     a[0][2] = 19;
22     y[0] = 81;
23     a[1][0] = 145;
24     a[1][1] = -134;
25     a[1][2] = 99;
26     y[1] = 12;
27     a[2][0] = 325;
28     a[2][1] = 991;
29     a[2][2] = -199;
30     y[2] = 213;
31 }
32
33
34
35 void write_data(){
36     int i;
37     for (i=0; i<n; i++)
38          printf("%lf ", coef[i]);
```

```
39      printf("\n");
40  }
41
42  double deter(double** a){
43      return(a[0][0]*(a[1][1]*a[2][2]-a[2][1]*a[1][2])
44                  -a[0][1]*(a[1][0]*a[2][2]-a[2][0]*a[1][2])
45                  +a[0][2]*(a[1][0]*a[2][1]-a[2][0]*a[1][1]));
46
47  }
48
50  void setup(double** b, double* coef, int j){
51      int i;
52      for(i=0; i<n; i++){
53          b[i][j] = y[i];
54          if (j>0)
55              b[i][j-1] = a[i][j-1];
56      }
57      coef[j] = deter(b) / det;
58  }
59
60  void solve(){
61      double** b;
62      b = (double**)malloc(RMAX*sizeof(double*));
63      int i,j;
64      for (i=0; i<RMAX; i++)
65          b[i] = (double*)malloc(CMAX*sizeof(double));
66
67      oshibka = 1;
68      for(i=0; i<n; i++)
69          for(j=0; j<n; j++)
70              b[i][j] = a[i][j];
71      det = deter(b);
72      if (det==0){
73          oshibka = 0;
74          printf("ERROR: matrix is singular.");
75      } else {
76          setup(b, coef, 0);
77          setup(b, coef, 1);
78          setup(b, coef, 2);
79      }
80  }
```

```
81
82 int main()
83 {
84     y = (double*)malloc(CMAX*sizeof(double));
85     coef = (double*)malloc(CMAX*sizeof(double));
86
87     a = (double**)malloc(RMAX*sizeof(double*));
88     int i;
89     for (i=0; i<RMAX; i++)
90         a[i] = (double*)malloc(CMAX*sizeof(double));
91
92     printf("\n");
93     printf("Simultaneous soulution by Cramers rule");
94     get_data(a, y);
95     SAMPLE;
96     solve();
97     SAMPLE;
98     if (oshibka == 0)
99         write_data();
100    printf("\n");
101    return 0;
102 }
```

ПРИЛОЖЕНИЕ В

ДЛЯ ВРЕМЕНИ ФУ

```
1  #include "Sampler.sh"
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define RMAX 3
6  #define CMAX 3
7
8  int n;
9  char yesno;
10 int oshibka;
11
12 double* y;
13 double* coef;
14 double** a;
15
16 double det;
17
18 void get_data(double** a, double* y){
19     a[0][0] = 1;
20     a[0][1] = -43;
21     a[0][2] = 19;
22     y[0] = 81;
23     a[1][0] = 145;
24     a[1][1] = -134;
25     a[1][2] = 99;
26     y[1] = 12;
27     a[2][0] = 325;
28     a[2][1] = 991;
29     a[2][2] = -199;
30     y[2] = 213;
31 }
32
33
34
35 void write_data(){
36     int i;
37     for (i=0; i<n; i++)
38          printf("%lf ", coef[i]);
```

```
39     printf("\n");
40 }
41
42 double deter(double** a){
43     return(a[0][0]*(a[1][1]*a[2][2]-a[2][1]*a[1][2])
44                 -a[0][1]*(a[1][0]*a[2][2]-a[2][0]*a[1][2])
45                 +a[0][2]*(a[1][0]*a[2][1]-a[2][0]*a[1][1]));
46
47 }
48
49 void setup(double** b, double* coef, int j){
50     int i;
51     for(i=0; i<n; i++){
52         b[i][j] = y[i];
53         if (j>0)
54             b[i][j-1] = a[i][j-1];
55     }
56     coef[j] = deter(b) / det;
57 }
58
59 void solve(){
60     SAMPLE;
61     double** b;
62     b = (double**)malloc(RMAX*sizeof(double*));
63     int i,j;
64     SAMPLE;
65     for (i=0; i<RMAX; i++)
66         b[i] = (double*)malloc(CMAX*sizeof(double));
67     SAMPLE;
68     oshibka = 1;
69     SAMPLE;
70    for(i=0; i<n; i++)
71         for(j=0; j<n; j++)
72             b[i][j] = a[i][j];
73     SAMPLE;
74     det = deter(b);
75     SAMPLE;
76    if (det==0){
77         oshibka = 0;
78         printf("ERROR: matrix is singular.");
79     SAMPLE;
```

```
80      } else {
81      SAMPLE;
82          setup(b, coef, 0);
83      SAMPLE;
84          setup(b, coef, 1);
85      SAMPLE;
86          setup(b, coef, 2);
87      }
88      SAMPLE;
89  }
90
91  int main()
92  {
93      y = (double*)malloc(CMAX*sizeof(double));
94      coef = (double*)malloc(CMAX*sizeof(double));
95
96      a = (double**)malloc(RMAX*sizeof(double*));
97      int i;
98      for (i=0; i<RMAX; i++)
99          a[i] = (double*)malloc(CMAX*sizeof(double));
100
101     printf("\n");
102     printf("Simultaneous soulution by Cramers rule");
103     get_data(a, y);
104     solve();
105     if (oshibka == 0)
106         write_data();
107     printf("\n");
108     return 0;
109 }
```