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 Рыбин А.С.

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'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

Выполняется **вариант 14**

Использовался **старый** *SAMPLER*, т.к. новый не работает на используемой виртуальной машине с *Microsoft Windows XP Professional 2002 Service Pack 3*.

Для компиляции программ использовался *Borland C++ 3.1*.

Профилирование тестовой программы **TEST_CYC.CPP**. Исходный код представлен в приложении А.

```
       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 :   8  1 :  10            4335.47            1            4335.47
-----------------------------------------------------------------------
 1 :  10  1 :  12            8670.11            1            8670.11
-----------------------------------------------------------------------
 1 :  12  1 :  14           21671.50            1           21671.50
-----------------------------------------------------------------------
 1 :  14  1 :  16           43348.03            1           43348.03
-----------------------------------------------------------------------
 1 :  16  1 :  19            4334.64            1            4334.64
-----------------------------------------------------------------------
 1 :  19  1 :  22            8670.11            1            8670.11
-----------------------------------------------------------------------
 1 :  22  1 :  25           21677.37            1           21677.37
-----------------------------------------------------------------------
 1 :  25  1 :  28           43343.00            1           43343.00
-----------------------------------------------------------------------
 1 :  28  1 :  34            4339.66            1            4339.66
-----------------------------------------------------------------------
 1 :  34  1 :  40            8670.11            1            8670.11
-----------------------------------------------------------------------
 1 :  40  1 :  46           21671.50            1           21671.50
-----------------------------------------------------------------------
 1 :  46  1 :  52           43348.03            1           43348.03
-----------------------------------------------------------------------
```

В результате профилирования видно, что полученные данные сильно завышены, что может объясняться накладными расходами эмулятора.

В программе используется разная запись циклов с одинаковым количеством итераций. По результатам профилирования заметно, что это не влияет на результат.

Также видна линейная зависимость времени выполнения от количества итераций.

Профилирование тестовой программы **TEST_SUB.CPP**. Исходный код представлен в приложении Б.

```
        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 :  30  1 :  32           433700.70             1          433700.70
-----------------------------------------------------------------------
 1 :  32  1 :  34           867387.15             1          867387.15
-----------------------------------------------------------------------
 1 :  34  1 :  36          2168474.16             1         2168474.16
-----------------------------------------------------------------------
 1 :  36  1 :  38          4336937.43             1         4336937.43
-----------------------------------------------------------------------
```

По результатам профилирования можно сделать аналогичные выводы о завышении результатов из-за использования эмулятора и линейной зависимости между временем выполнения цикла и количеством итераций в нём.

Профилирование полного времени выполнения программы из первой лабораторной работы. Исходный код представлен в приложении В (**LINFIT_FULL.CPP**).

```
      Отчет о результатах измерений для программы LINFIT~1.EXE.

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


            Список обработанных файлов.

-----------------------------------------------------------------------
   NN            Имя обработанного файла
-----------------------------------------------------------------------
   1.  LINFIT~1.CPP
-----------------------------------------------------------------------


   Таблица с результатами измерений ( используется 2 из 416 записей )

-----------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-----------------------------------------------------------------------
 1 :  47  1 :  56           57681.15            1          57681.15
-----------------------------------------------------------------------
```

Общее время выполнения программы **57681 мкс**.

Профилирование времени выполнения функциональных участков программы из первой лабораторной работы. Исходный код представлен в приложении Г (**LINFIT_FUN.CPP**).

```
      Отчет о результатах измерений для программы LINFIT~4.EXE.

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


            Список обработанных файлов.

-----------------------------------------------------------------------
   NN            Имя обработанного файла
-----------------------------------------------------------------------
   1.  LINFIT~4.CPP
-----------------------------------------------------------------------


   Таблица с результатами измерений ( используется 14 из 416 записей )

-----------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-----------------------------------------------------------------------
 1 :   8  1 :  19               4.19            1               4.19
-----------------------------------------------------------------------
 1 :  19  1 :  22               2.51            1               2.51
-----------------------------------------------------------------------
 1 :  22  1 :  34           17765.97         1000              17.77
-----------------------------------------------------------------------
 1 :  34  1 :  22            1754.97          999               1.76
 1 :  34  1 :  37               2.51            1               2.51
-----------------------------------------------------------------------
```

```
 1 :  37  1 :  45             470.17            1             470.17
-------------------------------------------------------------------------
 1 :  45  1 :  48               1.68            1               1.68
-------------------------------------------------------------------------
 1 :  48  1 :  52            8196.58         1000               8.20
-------------------------------------------------------------------------
 1 :  52  1 :  48            1765.87          999               1.77
 1 :  52  1 :  55               1.68            1               1.68
-------------------------------------------------------------------------
 1 :  55  1 :  81               2.51            1               2.51
-------------------------------------------------------------------------
 1 :  65  1 :  69               2.51            1               2.51
-------------------------------------------------------------------------
 1 :  69  1 :  74           30869.61         1000              30.87
-------------------------------------------------------------------------
 1 :  74  1 :  69            1758.33          999               1.76
 1 :  74  1 :  77               1.68            1               1.68
-------------------------------------------------------------------------
 1 :  77  1 :   8               7.54            1               7.54
-------------------------------------------------------------------------
```

Сумма времени выполнения функциональных участков почти совпадает с полным временем выполнения программы из предыдущего пункта (с разницей **5 мс**), что можно объяснить погрешностями и накладными расходами монитора за счёт большего количества измерений.

Больше всего времени уходит на циклы в строчках **22:34**, **48:52** и **69:74**.

Время в цикле *48:52* можно сократить за счёт уменьшения операций косвенного обращения к памяти (разыменования указателей *a* и *b*) с помощью использования дополнительных локальных переменных.

В цикле *69:74* можно убрать операции деления с остатком и вычитания, т.к. требуется заполнить массивы случайными данными и неважно из какого диапазона на результаты профилирования это не повлияет.

Цикл *22:34* можно было бы ускорить за счёт использования векторных инструкций, но они не доступны в такой ранней версии компилятора.

Профилирование полного времени выполнения оптимизированной версии программы из первой лабораторной работы. Исходный код представлен в приложении Д (**LINFIT_FULL_OPT.CPP**).

```
Отчет о результатах измерений для программы LINFIT~3.EXE.

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


Список обработанных файлов.
```

```
------------------------------------------------------------------------
   NN            Имя обработанного файла
------------------------------------------------------------------------
   1.  LINFIT~3.CPP
------------------------------------------------------------------------


   Таблица с результатами измерений ( используется 2 из 416 записей )

------------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
------------------------------------------------------------------------
 1 :  50  1 :  59           54684.12            1          54684.12
------------------------------------------------------------------------
```

Общее время выполнения программы **54684 мкс**, что на **3 мс** быстрее предыдущего варианта.

Профилирование времени выполнения функциональных участков оптимизированной версии программы из первой лабораторной работы. Исходный код представлен в приложении Е (**LINFIT_FUN_OPT.CPP**).

```
      Отчет о результатах измерений для программы LINFIT~2.EXE.

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


            Список обработанных файлов.

------------------------------------------------------------------------
   NN            Имя обработанного файла
------------------------------------------------------------------------
   1.  LINFIT~2.CPP
------------------------------------------------------------------------


   Таблица с результатами измерений ( используется 15 из 416 записей )

------------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
------------------------------------------------------------------------
 1 :   8  1 :  19              3.35            1               3.35
------------------------------------------------------------------------
 1 :  19  1 :  22              2.51            1               2.51
------------------------------------------------------------------------
 1 :  22  1 :  34          17771.84         1000              17.77
------------------------------------------------------------------------
 1 :  34  1 :  22           1783.47          999               1.79
 1 :  34  1 :  37              2.51            1               2.51
------------------------------------------------------------------------
 1 :  37  1 :  45            468.50            1             468.50
------------------------------------------------------------------------
 1 :  45  1 :  48              2.51            1               2.51
------------------------------------------------------------------------
 1 :  48  1 :  52           7500.13         1000               7.50
------------------------------------------------------------------------
```

```
 1 :  52  1 :  48            1762.52          999              1.76
 1 :  52  1 :  55               2.51            1              2.51
-------------------------------------------------------------------
 1 :  55  1 :  60               4.19            1              4.19
-------------------------------------------------------------------
 1 :  60  1 :  85               2.51            1              2.51
-------------------------------------------------------------------
 1 :  70  1 :  73               2.51            1              2.51
-------------------------------------------------------------------
 1 :  73  1 :  78           28219.55         1000             28.22
-------------------------------------------------------------------
 1 :  78  1 :  73            1765.03          999              1.77
 1 :  78  1 :  81               2.51            1              2.51
-------------------------------------------------------------------
 1 :  81  1 :   8               6.70            1              6.70
-------------------------------------------------------------------
```

Сумма времени выполнения функциональных участков почти совпадает с полным временем выполнения программы из предыдущего пункта (с разницей **5 мс**), что можно объяснить погрешностями и накладными расходами монитора за счёт большего количества измерений.

Среднее время одной итерации в цикле *48:52* уменьшилось на **0.3 мкс**.

Среднее время одной итерации в цикле *69:74* уменьшилось на **2.65 мкс**.

Выводы

В ходе выполнения лабораторной работы был изучен монитор для профилирования **SAMPLER**. С его мощью было выполнено профилирование тестовых программ *TEST_CYC.CPP* и *TEST_SUB.CPP*. В результате было установлено, что время выполнения цикла не зависит от его записи и линейно зависит от количества итераций.

Были получены профили программы из первой лабораторной работы в двух режимах: полное время выполнения и время выполнения функциональных участков. В результате были выявлены узкие места и выполнены оптимизации. После чего повторно были получены профили в двух режимах. В результате применения оптимизаций среднее время выполнения одной итерации в цикле *48:52* уменьшилось на **0.3 мкс**, в цикле *69:74* уменьшилось на **2.65 мкс**.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД TEST_CYC.CPP

```
01: #include <SAMPLER.H>
02:
03: #define Size 10000
04: int i, tmp, dim[Size];
05:
06: void main()
07: {
08:      SAMPLE;
09:      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: }
54:
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД TEST_SUB.CPP

```
01: #include <SAMPLER.H>
02:
03: const unsigned Size = 1000;
04:
05: void TestLoop(int nTimes)
06: {
07:   static int TestDim[Size];
08:   int tmp;
09:   int iLoop;
10:
11:   while (nTimes > 0)
12:   {
13:     nTimes --;
14:
15:     iLoop = Size;
16:     while (iLoop > 0)
17:     {
18:       iLoop -- ;
19:       tmp = TestDim[0];
20:       TestDim[0] = TestDim[nTimes];
21:       TestDim[nTimes] = tmp;
22:     }
23:   }
24: } /* TestLoop */
25:
27: void main()
28: {
29:    SAMPLE;
30:    TestLoop(Size / 10); // 100 * 1000  повторений
31:    SAMPLE;
32:    TestLoop(Size / 5);  // 200 * 1000  повторений
33:    SAMPLE;
34:    TestLoop(Size / 2);  // 500 * 1000  повторений
35:    SAMPLE;
36:    TestLoop(Size / 1);  // 1000* 1000  повторений
37:    SAMPLE;
38: }
39:
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД LINFIT_FULL.CPP

```
01: #include <stdlib.h>
02:
03: #include <SAMPLER.H>
04:
05: #define SIZE 1000
06:
07: void linfit(const float* x, const float* y, float* y_calc, float* a, float*
b, int n) {
08:     float sum_x, sum_y, sum_xy, sum_x2, xi, yi, sxx, sxy;
09:
10:     sum_x = 0.0;
11:     sum_y = 0.0;
12:     sum_xy = 0.0;
13:     sum_x2 = 0.0;
14:
15:     int i = 0;
16:
17:     for (i = 0; i < n; i++) {
18:         xi = x[i];
19:         yi = y[i];
20:
21:         sum_x += xi;
22:         sum_y += yi;
23:
24:         sum_xy += xi * yi;
25:
26:         sum_x2 += xi * xi;
27:     }
28:
29:     sxx = sum_x2 - sum_x * sum_x / n;
30:     sxy = sum_xy - sum_x * sum_y / n;
31:
32:     *a = sxy / sxx;
33:     *b = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
34:
35:     for (i = 0; i < n; i++) {
36:         y_calc[i] = *a + *b * x[i];
37:     }
```

```
38: }
39:
40: int main(int argc, char* argv[]) {
41:     float x[SIZE];
42:     float y[SIZE];
43:     float y_calc[SIZE];
44:
45:     float a, b;
46:
47:     SAMPLE;
48:
49:     for (int i = 0; i < SIZE; i++) {
50:         x[i] = rand() % 200 - 100; // [-100, 100)
51:         y[i] = rand() % 200 - 100; // [-100, 100)
52:     }
53:
54:     linfit(x, y, y_calc, &a, &b, SIZE);
55:
56:     SAMPLE;
57:
58:     return 0;
59: }
60:
```

ПРИЛОЖЕНИЕ Г

ИСХОДНЫЙ КОД LINFIT_FUN.CPP

```
01: #include <stdlib.h>
02:
03: #include <SAMPLER.H>
04:
05: #define SIZE 1000
06:
07: void linfit(const float* x, const float* y, float* y_calc, float* a, float*
b, int n) {
08:     SAMPLE;
09:
10:     float sum_x, sum_y, sum_xy, sum_x2, xi, yi, sxx, sxy;
11:
12:     sum_x = 0.0;
13:     sum_y = 0.0;
14:     sum_xy = 0.0;
15:     sum_x2 = 0.0;
16:
17:     int i = 0;
18:
19:     SAMPLE;
20:
21:     for (i = 0; i < n; i++) {
22:         SAMPLE;
23:
24:         xi = x[i];
25:         yi = y[i];
26:
27:         sum_x += xi;
28:         sum_y += yi;
29:
30:         sum_xy += xi * yi;
31:
32:         sum_x2 += xi * xi;
33:
34:         SAMPLE;
35:     }
36:
37:     SAMPLE;
```

```
38:
39:     sxx = sum_x2 - sum_x * sum_x / n;
40:     sxy = sum_xy - sum_x * sum_y / n;
41:
42:     *a = sxy / sxx;
43:     *b = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
44:
45:     SAMPLE;
46:
47:     for (i = 0; i < n; i++) {
48:         SAMPLE;
49:
50:         y_calc[i] = *a + *b * x[i];
51:
52:         SAMPLE;
53:     }
54:
55:     SAMPLE;
56: }
57:
58: int main(int argc, char* argv[]) {
59:     float x[SIZE];
60:     float y[SIZE];
61:     float y_calc[SIZE];
62:
63:     float a, b;
64:
65:     SAMPLE;
66:
67:     for (int i = 0; i < SIZE; i++) {
68:
69:         SAMPLE;
70:
71:         x[i] = rand() % 200 - 100; // [-100, 100)
72:         y[i] = rand() % 200 - 100; // [-100, 100)
73:
74:         SAMPLE;
75:     }
76:
77:     SAMPLE;
78:
```

```
79:     linfit(x, y, y_calc, &a, &b, SIZE);
80:
81:     SAMPLE;
82:
83:     return 0;
84: }
85:
```

ПРИЛОЖЕНИЕ Д

ИСХОДНЫЙ КОД LINFIT_FULL_OPT.CPP

```
01: #include <stdlib.h>
02:
03: #include <SAMPLER.H>
04:
05: #define SIZE 1000
06:
07: void linfit(const float *x, const float *y, float* y_calc, float* a, float*
b, int n) {
08:     float sum_x, sum_y, sum_xy, sum_x2, xi, yi, sxx, sxy, _a, _b;
09:
10:     sum_x = 0.0;
11:     sum_y = 0.0;
12:     sum_xy = 0.0;
13:     sum_x2 = 0.0;
14:
15:     int i = 0;
16:
17:     for (i = 0; i < n; i++) {
18:         xi = x[i];
19:         yi = y[i];
20:
21:         sum_x += xi;
22:         sum_y += yi;
23:
24:         sum_xy += xi * yi;
25:
26:         sum_x2 += xi * xi;
27:     }
28:
29:     sxx = sum_x2 - sum_x * sum_x / n;
30:     sxy = sum_xy - sum_x * sum_y / n;
31:
32:     _a = sxy / sxx;
33:     _b = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
34:
35:     for (i = 0; i < n; i++) {
36:         y_calc[i] = _a + _b * x[i];
37:     }
```

```
38:
39:     *a = _a;
40:     *b = _b;
41: }
42:
43: int main(int argc, char *argv[]) {
44:     float x[SIZE];
45:     float y[SIZE];
46:     float y_calc[SIZE];
47:
48:     float a, b;
49:
50:     SAMPLE;
51:
52:     for (int i = 0; i < SIZE; i++) {
53:         x[i] = rand();
54:         y[i] = rand();
55:     }
56:
57:     linfit(x, y, y_calc, &a, &b, SIZE);
58:
59:     SAMPLE;
60:
61:     return 0;
62: }
63:
```

ПРИЛОЖЕНИЕ Е

ИСХОДНЫЙ КОД LINFIT_FUN_OPT.CPP

```
01: #include <stdlib.h>
02:
03: #include <SAMPLER.H>
04:
05: #define SIZE 1000
06:
07: void linfit(const float *x, const float *y, float* y_calc, float* a, float*
b, int n) {
08:     SAMPLE;
09:
10:     float sum_x, sum_y, sum_xy, sum_x2, xi, yi, sxx, sxy, _a, _b;
11:
12:     sum_x = 0.0;
13:     sum_y = 0.0;
14:     sum_xy = 0.0;
15:     sum_x2 = 0.0;
16:
17:     int i = 0;
18:
19:     SAMPLE;
20:
21:     for (i = 0; i < n; i++) {
22:         SAMPLE;
23:
24:         xi = x[i];
25:         yi = y[i];
26:
27:         sum_x += xi;
28:         sum_y += yi;
29:
30:         sum_xy += xi * yi;
31:
32:         sum_x2 += xi * xi;
33:
34:         SAMPLE;
35:     }
36:
37:     SAMPLE;
```

```
38:
39:     sxx = sum_x2 - sum_x * sum_x / n;
40:     sxy = sum_xy - sum_x * sum_y / n;
41:
42:     _a = sxy / sxx;
43:     _b = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
44:
45:     SAMPLE;
46:
47:     for (i = 0; i < n; i++) {
48:         SAMPLE;
49:
50:         y_calc[i] = _a + _b * x[i];
51:
52:         SAMPLE;
53:     }
54:
55:     SAMPLE;
56:
57:     *a = _a;
58:     *b = _b;
59:
60:     SAMPLE;
61: }
62:
63: int main(int argc, char *argv[]) {
64:     float x[SIZE];
65:     float y[SIZE];
66:     float y_calc[SIZE];
67:
68:     float a, b;
69:
70:     SAMPLE;
71:
72:     for (int i = 0; i < SIZE; i++) {
73:         SAMPLE;
74:
75:         x[i] = rand();
76:         y[i] = rand();
77:
78:         SAMPLE;
```

```
79:     }
80:
81:     SAMPLE;
82:
83:     linfit(x, y, y_calc, &a, &b, SIZE);
84:
85:     SAMPLE;
86:
87:     return 0;
88: }
89:
```