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 |  | Путьков Д.А. |
| 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 c 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.

1. 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.

**Тестовые программы**

Код программы test\_cyc.c 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 видно, что времена сильно завышены из-за накладных затрат эмулятора. В коде используется разная запись циклов с одинаковым количеством итераций, при этом отсутствует влияние на время. А также видна линейная зависимость времени от количества итераций.

Код программы test\_sub.c с нумерацией строк представлен в приложении Б.

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 можно сделать аналогичные выводы о том, что время выполнения:

1. линейно зависит от количества итераций цикла;
2. сильно завышено из-за накладных затрат эмулятора.

**Программа из первой лабораторной работы**

Код программы из первой лабораторной работы с нумерацией строк представлен в приложениях В (для измерения полного времени) и Г (для измерения времен выполнения ФУ).

Результаты профилирования с измерением полного времени:

----------------------------------------------------------------------

NN Имя обработанного файла

----------------------------------------------------------------------

1. LR1\_1.CPP

----------------------------------------------------------------------

Таблица с результатами измерений ( используется 3 из 416 записей )

----------------------------------------------------------------------

Исх.Поз. Прием.Поз. Общее время(мкс) Кол-во прох. Среднее время(мкс)

---------------------------------------------------------------------- 1 : 45 1 : 47 5198277.71 1 5198277.71

---------------------------------------------------------------------- 1 : 47 1 : 49 11125060.76 1 11125060.76

Общее время выполнения первой функции – 5198278 мкс, второй – практически в два раза больше и составляет 11125061 мкс. Результаты также завышены из-за накладных затрат эмулятора.

Результаты профилирования с измерением времен ФУ:

----------------------------------------------------------------------

NN Имя обработанного файла

----------------------------------------------------------------------

1. LR1\_2.CPP

----------------------------------------------------------------------

Таблица с результатами измерений ( используется 17 из 416 записей )

----------------------------------------------------------------------

Исх.Поз. Прием.Поз. Общее время(мкс) Кол-во прох. Среднее время(мкс)

---------------------------------------------------------------------- 1 : 9 1 : 11 3.35 1 3.35

---------------------------------------------------------------------- 1 : 11 1 : 13 3438.71 999 3.44

----------------------------------------------------------------------

|  |  |  |  |
| --- | --- | --- | --- |
| 1 : 13 1 : 15 | 6797649.14 | 46081 | 147.52 |
| 1 : 13 1 : 21 | 11933956.32 | 60203 | 198.23 |

---------------------------------------------------------------------- 1 : 15 1 : 19 7090481.74 46081 153.87

----------------------------------------------------------------------

1 : 19 1 : 21 5436104.36 46081 117.97

|  |  |  |  |
| --- | --- | --- | --- |
| ---------------------------------------------------------------------- | | | |
| 1 : 21 1 : 13 | 19257288.09 | 39749 | 484.47 |
| 1 : 21 1 : 23 | 2106.14 | 999 | 2.11 |
| ---------------------------------------------------------------------- | | | |
| 1 : 23 1 : 11 | 57690.37 | 998 | 57.81 |
| 1 : 23 1 : 25 | 2.51 | 1 | 2.51 |

---------------------------------------------------------------------- 1 : 25 1 : 34 6.70 1 6.70

---------------------------------------------------------------------- 1 : 34 1 : 36 1.68 1 1.68

---------------------------------------------------------------------- 1 : 36 1 : 38 1076.12 972 1.11

---------------------------------------------------------------------- 1 : 38 1 : 40 2699.51 972 2.78

----------------------------------------------------------------------

|  |  |  |  |
| --- | --- | --- | --- |
| 1 : 40 1 : 42 | 9648944.84 | 48210 | 200.14 |
| 1 : 40 1 : 47 | 34795041.82 | 5314 | 6547.81 |

---------------------------------------------------------------------- 1 : 42 1 : 45 11634988.02 48210 241.34

---------------------------------------------------------------------- 1 : 45 1 : 47 8151851.36 48210 169.09

----------------------------------------------------------------------

|  |  |  |  |
| --- | --- | --- | --- |
| 1 : 47 1 : 40 | 40013272.10 | 52552 | 761.40 |
| 1 : 47 1 : 49 | 2354.21 | 972 | 2.42 |
| ---------------------------------------------------------------------- | | | |
| 1 : 49 1 : 36 | 1734.86 | 971 | 1.79 |
| 1 : 49 1 : 51 | 1.68 | 1 | 1.68 |

По результатам измерений времени на ФУ видно, что время выполнения первой функции – 50578720.73 мкс, второй – 104251966.2 мкс. Данные времена отличаются от полученных ранее примерно в 10 раз, что также может быть вызвано использованием эмулятора, однако, время выполнения второй функции по-прежнему в 2 раза больше первой. Одной из причин является вызов функции *swap* внутри второй функции. Для усовершенствования производительности заменим вызов функции на её содержимое.

**Измененная программа из первой лабораторной работы**

Измененный код программы из первой лабораторной работы с нумерацией строк представлен в приложениях Д (для измерения полного времени) и Е (для измерения времен выполнения ФУ).

Результаты профилирования с измерением полного времени:

----------------------------------------------------------------------

NN Имя обработанного файла

----------------------------------------------------------------------

* 1. LR1\_3.CPP

----------------------------------------------------------------------

Таблица с результатами измерений ( используется 3 из 416 записей )

----------------------------------------------------------------------

Исх.Поз. Прием.Поз. Общее время(мкс) Кол-во прох. Среднее время(мкс)

---------------------------------------------------------------------- 1 : 42 1 : 44 5180792.50 1 5180792.50

---------------------------------------------------------------------- 1 : 44 1 : 46 9568359.34 1 9568359.34

----------------------------------------------------------------------

Общее время выполнения первой функции не изменилось и составляет 5180792 мкс, время второй функции уменьшилось после изменений примерно на 15% и составляет 9568359 мкс. Результаты также завышены из-за накладных затрат эмулятора.

Результаты профилирования с измерением времен ФУ:

----------------------------------------------------------------------

NN Имя обработанного файла

----------------------------------------------------------------------

1. LR1\_4.CPP

----------------------------------------------------------------------

Таблица с результатами измерений ( используется 17 из 416 записей )

----------------------------------------------------------------------

Исх.Поз. Прием.Поз. Общее время(мкс) Кол-во прох. Среднее время(мкс)

---------------------------------------------------------------------- 1 : 9 1 : 11 2.51 1 2.51

---------------------------------------------------------------------- 1 : 11 1 : 13 2595.58 999 2.60

----------------------------------------------------------------------

|  |  |  |  |
| --- | --- | --- | --- |
| 1 : 13 1 : 21 | 12578211.17 | 56647 | 222.05 |
| 1 : 13 1 : 15 | 6778565.68 | 49637 | 136.56 |

---------------------------------------------------------------------- 1 : 15 1 : 19 6920225.78 49637 139.42

---------------------------------------------------------------------- 1 : 19 1 : 21 5534971.25 49637 111.51

----------------------------------------------------------------------

|  |  |  |  |
| --- | --- | --- | --- |
| 1 : 21 1 : 13 | 20130039.06 | 39749 | 506.43 |
| 1 : 21 1 : 23 | 1252.12 | 999 | 1.25 |
| ---------------------------------------------------------------------- | | | |
| 1 : 23 1 : 11 | 1934.33 | 998 | 1.94 |
| 1 : 23 1 : 25 | 2.51 | 1 | 2.51 |

---------------------------------------------------------------------- 1 : 25 1 : 29 5.87 1 5.87

---------------------------------------------------------------------- 1 : 29 1 : 31 0.84 1 0.84

---------------------------------------------------------------------- 1 : 31 1 : 33 257.30 915 0.28

---------------------------------------------------------------------- 1 : 33 1 : 35 2068.42 915 2.26

----------------------------------------------------------------------

|  |  |  |  |
| --- | --- | --- | --- |
| 1 : 35 1 : 44 | 31161548.13 | 19253 | 1618.53 |
| 1 : 35 1 : 37 | 9548283.58 | 42864 | 222.76 |

---------------------------------------------------------------------- 1 : 37 1 : 42 10427956.39 42864 243.28

---------------------------------------------------------------------- 1 : 42 1 : 44 7947496.61 42864 185.41

|  |  |  |  |
| --- | --- | --- | --- |
| ---------------------------------------------------------------------- | | | |
| 1 : 44 1 : 35 | 37930926.60 | 61202 | 619.77 |
| 1 : 44 1 : 46 | 1462.48 | 915 | 1.60 |
| ---------------------------------------------------------------------- | | | |
| 1 : 46 1 : 31 | 830.55 | 914 | 0.91 |
| 1 : 46 1 : 48 | 0.84 | 1 | 0.84 |

По результатам измерений времени на ФУ видно, что время выполнения первой функции – 51948200 мкс, второй – 97020832 мкс. В результате можно сделать аналогичные выводы, что и для измерений полного времени.

**Выводы**

В результате выполнения данной лабораторной работы был изучен монитор SAMPLER, с помощью которого было выполнено профилирование тестовых программ test\_cyc.c и test\_sub.c.

Было проанализировано полное время выполнения программы, разработанной в 1-ой лабораторной работе, и время выполнения её ФУ.

Удалось частично усовершенствовать производительность программы из 1-ой лабораторной работы за счёт удаления внутреннего вызова функции *swap*.

**ПРИЛОЖЕНИЕ А TEST\_CYC.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 Б TEST\_SUB.C

1. #include <stdlib.h>
2. #include "Sample.h"
3. const unsigned Size = 1000; 4

5

6 void TestLoop(int nTimes) 7 {

1. static int TestDim[Size];
2. int tmp;
3. int iLoop;

11

12 while (nTimes > 0) 13 {

14 nTimes --; 15

1. iLoop = Size;
2. while (iLoop > 0) 18 {
3. iLoop -- ;
4. tmp = TestDim[0];
5. TestDim[0] = TestDim[nTimes];
6. TestDim[nTimes] = tmp; 23 }

24 }

25 } /\* TestLoop \*/ 26

27

28 void main() 29 {

30 SAMPLE;

31 TestLoop(Size / 10); // 100 \* 1000 ¯®¢â®à¥•¨©

32 SAMPLE;

33 TestLoop(Size / 5); // 200 \* 1000 ¯®¢â®à¥•¨©

34 SAMPLE;

35 TestLoop(Size / 2); // 500 \* 1000 ¯®¢â®à¥•¨©

36 SAMPLE;

37 TestLoop(Size / 1); // 1000\* 1000 ¯®¢â®à¥•¨©

38 SAMPLE;

39 }

ПРИЛОЖЕНИЕ В

**Полное время LR\_1.C**

1. #include <math.h>
2. #include <stdlib.h>
3. #include <time.h>
4. #include <stdio.h>
5. #include "Sampler.h" 6
6. void sort1(float\* x, int n){
7. float hold;

9 for (int i = 0; i < n - 1; i++) {

10 for (int j = i + 1; j < n; j++) {

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 11 |  |  | if | (x[i] | > | x[j]) { |
| 12 |  |  |  | hold | = | x[i]; |
| 13 |  |  |  | x[i] | = | x[j]; |
| 14 |  |  |  | x[j] | = | hold; |
| 15 |  |  | } |  | | |
| 16 |  | } |  |
| 17 | } |  |  |

18 }

1. void swap(float \*a, float \*b) {
2. float hold = (\*a); 21 \*a = (\*b);

22 \*b = hold;

23 }

1. void sort2(float \*x, int n){
2. int no\_change = 0;
3. while(!no\_change) {
4. no\_change = 1;

28 for (int j=0; j < n-1; j++) { 29 if (x[j] > x[j+1]) {

30 swap(&x[j], &x[j+1]);

31 no\_change = 0;

32 }

33 }

34 }

35 }

36 int main(){

37 float x[1000];

38 float y[1000];

39

40 srand(time(NULL));

41 for (int i=0; i <1000; i++) { 42 x[i] = 1 + rand() % 999;

43 y[i] = x[i]; 44 }

45 SAMPLE;

46 sort1(x,1000);

47 SAMPLE;

48 sort2(y,1000);

1. SAMPLE;
2. return 0; 51 }

ПРИЛОЖЕНИЕ Г

**Время ФУ LR\_1.C**

1. #include <math.h>
2. #include <stdlib.h>
3. #include <time.h>
4. #include <stdio.h>
5. #include "Sampler.h" 6

7 void sort1(float\* x, int n){

|  |  |  |
| --- | --- | --- |
| 8 | float hold; |  |
| 9 | SAMPLE; |
| 10 | for (int i = 0; i < n - 1; i++) { |
| 11 | SAMPLE; |
| 12 | for (int j = i + 1; j < n; j++) | { |
| 13 | SAMPLE; |  |
| 14 | if (x[i] > x[j]) { |  |
| 15 | SAMPLE; |  |
| 16 | hold = x[i]; |  |
| 17 | x[i] = x[j]; |  |
| 18 | x[j] = hold; |  |
| 19 | SAMPLE; |  |
| 20 | } |  |
| 21 | SAMPLE; |  |
| 22 | } |  |
| 23 | SAMPLE; |  |
| 24 | } |  |
| 25 | SAMPLE; |  |
| 26 | } |  |
| 27 | void swap(float \*a, float \*b) { |  |
| 28 | float hold = (\*a); |  |
| 29 | \*a = (\*b); |  |
| 30 | \*b = hold; |  |
| 31 | } |  |
| 32 | void sort2(float \*x, int n){ |  |
| 33 | int no\_change = 0; |  |
| 34 | SAMPLE; |  |
| 35 | while(!no\_change) { |  |
| 36 | SAMPLE; |  |
| 37 | no\_change = 1; |  |
| 38 | SAMPLE; |  |
| 39 | for (int j=0; j < n-1; j++) { |  |
| 40 | SAMPLE; |  |
| 41 | if (x[j] > x[j+1]) { |  |
| 42 | SAMPLE; |  |
| 43 | swap(&x[j], &x[j+1]); |  |
| 44 | no\_change = 0; |  |
| 45 | SAMPLE; |  |
| 46 | } |  |
| 47 | SAMPLE; |  |
| 48 | } |  |
| 49 | SAMPLE; |  |
| 50 | } |  |
| 51 | SAMPLE; |  |
| 52 | } |  |
| 53 | int main(){ |  |
| 54 | float x[1000]; |  |
| 55 | float y[1000]; |  |
| 56 |  |  |
| 57 | srand(time(NULL)); |  |
| 58 | for (int i=0; i <1000; i++) { |  |

|  |  |  |
| --- | --- | --- |
| 59 |  | x[i] = 1 + rand() % 999; |
| 60 |  | y[i] = x[i]; |
| 61 |  | } |
| 62 |  | sort1(x,1000); |
| 63 |  | sort2(y,1000); |
| 64 |  | return 0; |
| 65 | } |  |

ПРИЛОЖЕНИЕ Д

**Полное время измененной LR\_1.C**

1. #include <math.h>
2. #include <stdlib.h>
3. #include <time.h>
4. #include <stdio.h>
5. #include "Sampler.h" 6
6. void sort1(float\* x, int n){
7. float hold;

9 for (int i = 0; i < n - 1; i++) {

10 for (int j = i + 1; j < n; j++) {

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 11 |  |  | if | (x[i] | > | x[j]) { | |
| 12 |  |  |  | hold | = | x[i]; | |
| 13 |  |  |  | x[i] | = | x[j]; | |
| 14 |  |  |  | x[j] | = | hold; | |
| 15 |  |  | } |  |  |  | |
| 16 |  | } |  |  |  |  | |
| 17 | } |  |  |  |  |  | |
| 18 | } | | | | | |  |
| 19 | void sort2(float \*x, int n){ | | | | | |  |
| 20 | int no\_change = 0; | | | | | |  |
| 21 | while(!no\_change) { | | | | | |  |
| 22 | no\_change = 1; | | | | | |  |
| 23 | for (int j=0; j < n-1; j++) | | | | | | { |
| 24 | if (x[j] > x[j+1]) { | | | | | |  |
| 25 | float hold = x[j]; | | | | | |  |
| 26 | x[j] = x[j + 1]; | | | | | |  |
| 27 | x[j + 1] = hold; | | | | | |  |
| 28 | no\_change = 0; | | | | | |  |
| 29 | } | | | | | |  |
| 30 | } | | | | | |  |
| 31 | } | | | | | |  |
| 32 | } | | | | | |  |
| 33 | int main(){ | | | | | |  |
| 34 | float x[1000]; | | | | | |  |
| 35 | float y[1000]; | | | | | |  |
| 36 |  | | | | | |  |
| 37 | srand(time(NULL)); | | | | | |  |
| 38 | for (int i=0; i <1000; i++) { | | | | | |  |
| 39 | x[i] = 1 + rand() % 999; | | | | | |  |
| 40 | y[i] = x[i]; | | | | | |  |
| 41 | } | | | | | |  |
| 42 | SAMPLE; | | | | | |  |
| 43 | sort1(x,1000); | | | | | |  |
| 44 | SAMPLE; | | | | | |  |
| 45 | sort2(y,1000); | | | | | |  |
| 46 | SAMPLE; | | | | | |  |
| 47 | return 0; | | | | | |  |
| 48 | } | | | | | |  |

ПРИЛОЖЕНИЕ Е

**Время ФУ измененной LR\_1.C**

1. #include <math.h>
2. #include <stdlib.h>
3. #include <time.h>
4. #include <stdio.h>
5. #include "Sampler.h" 6

7 void sort1(float\* x, int n){

|  |  |  |
| --- | --- | --- |
| 8 | float hold; |  |
| 9 | SAMPLE; |
| 10 | for (int i = 0; i < n - 1; i++) { |
| 11 | SAMPLE; |
| 12 | for (int j = i + 1; j < n; j++) | { |
| 13 | SAMPLE; |  |
| 14 | if (x[i] > x[j]) { |  |
| 15 | SAMPLE; |  |
| 16 | hold = x[i]; |  |
| 17 | x[i] = x[j]; |  |
| 18 | x[j] = hold; |  |
| 19 | SAMPLE; |  |
| 20 | } |  |
| 21 | SAMPLE; |  |
| 22 | } |  |
| 23 | SAMPLE; |  |
| 24 | } |  |
| 25 | SAMPLE; |  |
| 26 | } |  |
| 27 | void sort2(float \*x, int n){ |  |
| 28 | int no\_change = 0; |  |
| 29 | SAMPLE; |  |
| 30 | while(!no\_change) { |  |
| 31 | SAMPLE; |  |
| 32 | no\_change = 1; |  |
| 33 | SAMPLE; |  |
| 34 | for (int j=0; j < n-1; j++) { |  |
| 35 | SAMPLE; |  |
| 36 | if (x[j] > x[j+1]) { |  |
| 37 | SAMPLE; |  |
| 38 | float hold = x[j]; |  |
| 39 | x[j] = x[j + 1]; |  |
| 40 | x[j + 1] = hold; |  |
| 41 | no\_change = 0; |  |
| 42 | SAMPLE; |  |
| 43 | } |  |
| 44 | SAMPLE; |  |
| 45 | } |  |
| 46 | SAMPLE; |  |
| 47 | } |  |
| 48 | SAMPLE; |  |
| 49 | } |  |
| 50 | int main(){ |  |
| 51 | float x[1000]; |  |
| 52 | float y[1000]; |  |
| 53 |  |  |
| 54 | srand(time(NULL)); |  |
| 55 | for (int i=0; i <1000; i++) { |  |
| 56 | x[i] = 1 + rand() % 999; |  |
| 57 | y[i] = x[i]; |  |
| 58 | } |  |

|  |  |  |
| --- | --- | --- |
| 59 |  | sort1(x,1000); |
| 60 |  | sort2(y,1000); |
| 61 |  | return 0; |
| 62 | } |  |