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 _____ Тимофеев А.А.

Преподаватель _____ Кирьянчиков В.А.

Санкт-Петербург
2020

Задание.

1. Ознакомиться с документацией на 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.

3. Выявить "узкие места", ввести в программы 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

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 20	1 : 23	3782.33	1	3782.33
1 : 23	1 : 26	9453.73	1	9453.73
1 : 26	1 : 29	18904.1	1	18904.1
1 : 29	1 : 31	0.84	1	0.84
1 : 31	1 : 37	1892.42	1	1892.42
1 : 37	1 : 43	3780.65	1	3780.65
1 : 43	1 : 49	9452.05	1	9452.05
1 : 49	1 : 55	18905.78	1	18905.78

Была выявлена линейная зависимость результата от количества итераций цикла. Также видно, что времена сильно завышены из-за накладных затрат эмулятора.

Код программы test_sub.c с нумерацией строк представлен в приложении Б.

Результаты профилирования

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 23	1 : 25	6998.11	1	6998.11
1 : 25	1 : 27	13996.21	1	13996.21
1 : 27	1 : 29	34990.53	1	34990.53
1 : 29	1 : 31	69858.70	1	69858.70

Так как в процедуре выполняется достаточно большое количество итераций, можно проследить зависимость времени выполнения процедуры от количества итераций внутри процедуры. Линейная зависимость времени от количества итераций также прослеживается.

Программа из первой лабораторной работы

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

Результаты профилирования

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 10	1 : 16	144.15	1	144.15
1 : 16	1 : 26	15554.23	1	15554.23
1 : 26	1 : 36	979.73	1	979.73
1 : 36	1 : 40	7428.05	1	7428.05

Было произведено профилирование только вызова процедуры

Результаты профилирования

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 48	1 : 50	24111.20	1	24111.20

Сумма времён функциональных участков равна 24106.15 и почти равна результату профилирования только вызова процедуры. Выполнение процедуры занимает 24111мкс (0.024с).

Можно сделать оптимизацию – разыменовывание выполняется каждую итерацию цикла, хотя это можно сделать перед циклом. Применим оптимизацию.

Код оптимизированной программы из первой лабораторной работы с нумерацией строк представлен в приложениях Д и Е (для измерения полного времени и времени выполнения ФУ соответственно).

Результаты профилирования

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 10	1 : 16	142.48	1	142.48
1 : 16	1 : 26	15556.75	1	15556.75
1 : 26	1 : 36	978.06	1	978.06
1 : 36	1 : 41	6313.38	1	6313.38

Было произведено профилирование только вызова процедуры

Результаты профилирования

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 49	1 : 51	22997.37	1	22997.37

Сумма времён функциональных участков равна 22990.67 и почти равна результату профилирования только вызова процедуры.

Выполнение процедуры занимает 22997мкс (0.023с). Это на (24111 – 22997 = 1114мкс) быстрее, чем до оптимизации. Сегменты 10-16, 16-26, 26-36 выполняются за такое же время, однако время сегмента 36-41 значительно снизилось на (7428 – 6313 = 1115мкс). Это время тратилось на операцию разыменовывания на каждой итерации цикла вместо того, чтобы обращаться напрямую к данным. Оптимизация прошла успешно.

Вывод:

В ходе выполнения лабораторной работы был освоен профайлер Sampler. С его помощью была произведена профилировка четырёх программ, написанных на языке C. Производилось профилирование как для функциональных участков процедуры, так и только вызова процедуры. Удалось частично усовершенствовать производительность программы из 1-ой лабораторной работы за счёт выноса разыменовывания из цикла.

ПРИЛОЖЕНИЕ А

```
#include <Sampler.h>
#define Size 10000
int i, tmp, dim[Size];

void main()
{
   SAMPLE ;
      for(i=0;i<Size/10;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
   SAMPLE ;
      for(i=0;i<Size/5;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
   SAMPLE ;
      for(i=0;i<Size/2;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
   SAMPLE ;
      for(i=0;i<Size;i++){ tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
   SAMPLE ;

   SAMPLE ;
      for(i=0;i<Size/10;i++)
      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
   SAMPLE ;
      for(i=0;i<Size/5;i++)
      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
   SAMPLE ;
      for(i=0;i<Size/2;i++)
      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
   SAMPLE ;
      for(i=0;i<Size;i++)
      { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };
   SAMPLE ;

   SAMPLE ;
      for(i=0;i<Size/10;i++)
      { tmp=dim[0];
         dim[0]=dim[i];
         dim[i]=tmp;
      };
   SAMPLE ;
      for(i=0;i<Size/5;i++)
      { tmp=dim[0];
         dim[0]=dim[i];
         dim[i]=tmp;
      };
   SAMPLE ;
```

```
      for(i=0;i<Size/2;i++)
      { tmp=dim[0];
         dim[0]=dim[i];
         dim[i]=tmp;
      };
   SAMPLE ;
      for(i=0;i<Size;i++)
      { tmp=dim[0];
         dim[0]=dim[i];
         dim[i]=tmp;
      };
   SAMPLE ;

}
```

ПРИЛОЖЕНИЕ Б

```
#include <Sampler.h>
const unsigned Size = 1000;
void TestLoop(int nTimes)
{
  static int TestDim[Size];
  int tmp;
  int iLoop;
  while (nTimes > 0)
  {
    nTimes --;
    iLoop = Size;
    while (iLoop > 0)
    {
      iLoop -- ;
      tmp = TestDim[0];
      TestDim[0] = TestDim[nTimes];
      TestDim[nTimes] = tmp;
    }
  }
} /* TestLoop */
void main()
{
  SAMPLE;
  TestLoop(Size / 10); // 100 * 1000
  SAMPLE;
  TestLoop(Size / 5);  // 200 * 1000
  SAMPLE;
  TestLoop(Size / 2);  // 500 * 1000
  SAMPLE;
  TestLoop(Size / 1);  // 1000* 1000
  SAMPLE;
}

```

ПРИЛОЖЕНИЕ В

```
#include <math.h>
#include <Sampler.h>

void linfit2(float *x, float *y, float *y_calc,
             float *a, float *b, int n){
    int i;
    float sum_x, sum_y, sum_xy, sum_x2, sum_y2;
    float xi, yi, sxy, syy, sxx;
    float correl_coef, see, sigma_b, sigma_a;
    sum_x = 0.0;
    sum_y = 0.0;
    sum_xy = 0.0;
    sum_x2 = 0.0;
    sum_y2 = 0.0;
    for(i = 0; i < n; i++){
        xi = x[i];
        yi = y[i];
        sum_x = sum_x + xi;
        sum_y = sum_y + yi;
        sum_xy = sum_xy + xi * yi;
        sum_x2 = sum_x2 + xi * xi;
        sum_y2 = sum_y2 + yi * yi;
    }
    sxx = sum_x2 - sum_x * sum_x / n;
    sxy = sum_xy - sum_x * sum_y / n;
    syy = sum_y2 - sum_y * sum_y / n;
    *b = sxy / sxx;
    *a = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
    correl_coef = sxy / sqrt(sxx * syy);
    see = sqrt((sum_y2 - (*a) * sum_y - (*b) * sum_xy)/(n - 2));
    sigma_b = see / sqrt(sxx);
    sigma_a = sigma_b * sqrt(sum_x2 / n);
    for(i = 0; i < n; i++){
        y_calc[i] = (*a) + (*b) * x[i];
    }
}

int main(){
    float x[1000];
    float y[1000];
    float y_calc[1000];
    float a, b;
    int i;
    for(i = 0; i < 1000; i++){
        x[i] = i / 3.0;
        y[i] = i * i / 3.0;
    }
    SAMPLE;
    linfit2(x, y, y_calc, &a, &b, 1000);
    SAMPLE;
}
```

ПРИЛОЖЕНИЕ Г

```
#include <math.h>
#include <Sampler.h>

void linfit2(float *x, float *y, float *y_calc,
             float *a, float *b, int n){
    int i;
    float sum_x, sum_y, sum_xy, sum_x2, sum_y2;
    float xi, yi, sxy, syy, sxx;
    float correl_coef, see, sigma_b, sigma_a;
    SAMPLE;
    sum_x = 0.0;
    sum_y = 0.0;
    sum_xy = 0.0;
    sum_x2 = 0.0;
    sum_y2 = 0.0;
    SAMPLE;
    for(i = 0; i < n; i++){
        xi = x[i];
        yi = y[i];
        sum_x = sum_x + xi;
        sum_y = sum_y + yi;
        sum_xy = sum_xy + xi * yi;
        sum_x2 = sum_x2 + xi * xi;
        sum_y2 = sum_y2 + yi * yi;
    }
    SAMPLE;
    sxx = sum_x2 - sum_x * sum_x / n;
    sxy = sum_xy - sum_x * sum_y / n;
    syy = sum_y2 - sum_y * sum_y / n;
    *b = sxy / sxx;
    *a = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
    correl_coef = sxy / sqrt(sxx * syy);
    see = sqrt((sum_y2 - (*a) * sum_y - (*b) * sum_xy)/(n - 2));
    sigma_b = see / sqrt(sxx);
    sigma_a = sigma_b * sqrt(sum_x2 / n);
    SAMPLE;
    for(i = 0; i < n; i++){
        y_calc[i] = (*a) + (*b) * x[i];
    }
    SAMPLE;
}
```

```

int main(){
    float x[1000];
    float y[1000];
    float y_calc[1000];
    float a, b;
    int i;
    for(i = 0; i < 1000; i++){
        x[i] = i / 3.0;
        y[i] = i * i / 3.0;
    }
    linfit2(x, y, y_calc, &a, &b, 1000);
}

```

ПРИЛОЖЕНИЕ Д

```
#include <math.h>
#include <Sampler.h>

void linfit2(float *x, float *y, float *y_calc,
             float *a, float *b, int n){
    int i;
    float sum_x, sum_y, sum_xy, sum_x2, sum_y2;
    float xi, yi, sxy, syy, sxx;
    float correl_coef, see, sigma_b, sigma_a;
    sum_x = 0.0;
    sum_y = 0.0;
    sum_xy = 0.0;
    sum_x2 = 0.0;
    sum_y2 = 0.0;
    for(i = 0; i < n; i++){
        xi = x[i];
        yi = y[i];
        sum_x = sum_x + xi;
        sum_y = sum_y + yi;
        sum_xy = sum_xy + xi * yi;
        sum_x2 = sum_x2 + xi * xi;
        sum_y2 = sum_y2 + yi * yi;
    }
    sxx = sum_x2 - sum_x * sum_x / n;
    sxy = sum_xy - sum_x * sum_y / n;
    syy = sum_y2 - sum_y * sum_y / n;
    *b = sxy / sxx;
    *a = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
    correl_coef = sxy / sqrt(sxx * syy);
    see = sqrt((sum_y2 - (*a) * sum_y - (*b) * sum_xy)/(n - 2));
    sigma_b = see / sqrt(sxx);
    sigma_a = sigma_b * sqrt(sum_x2 / n);
    int c = *a, d = *b;
    for(i = 0; i < n; i++){
        y_calc[i] = c + d * x[i];
    }
}
```

```

int main(){
    float x[1000];
    float y[1000];
    float y_calc[1000];
    float a, b;
    int i;
    for(i = 0; i < 1000; i++){
        x[i] = i / 3.0;
        y[i] = i * i / 3.0;
    }
    SAMPLE;
    linfit2(x, y, y_calc, &a, &b, 1000);
    SAMPLE;
}

```

ПРИЛОЖЕНИЕ Е

```
#include <math.h>
#include <Sampler.h>

void linfit2(float *x, float *y, float *y_calc,
             float *a, float *b, int n){
    int i;
    float sum_x, sum_y, sum_xy, sum_x2, sum_y2;
    float xi, yi, sxy, syy, sxx;
    float correl_coef, see, sigma_b, sigma_a;
    SAMPLE;
    sum_x = 0.0;
    sum_y = 0.0;
    sum_xy = 0.0;
    sum_x2 = 0.0;
    sum_y2 = 0.0;
    SAMPLE;
    for(i = 0; i < n; i++){
        xi = x[i];
        yi = y[i];
        sum_x = sum_x + xi;
        sum_y = sum_y + yi;
        sum_xy = sum_xy + xi * yi;
        sum_x2 = sum_x2 + xi * xi;
        sum_y2 = sum_y2 + yi * yi;
    }
    SAMPLE;
    sxx = sum_x2 - sum_x * sum_x / n;
    sxy = sum_xy - sum_x * sum_y / n;
    syy = sum_y2 - sum_y * sum_y / n;
    *b = sxy / sxx;
    *a = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
    correl_coef = sxy / sqrt(sxx * syy);
    see = sqrt((sum_y2 - (*a) * sum_y - (*b) * sum_xy)/(n - 2));
    sigma_b = see / sqrt(sxx);
    sigma_a = sigma_b * sqrt(sum_x2 / n);
    SAMPLE;
    int c = *a, d = *b;
    for(i = 0; i < n; i++){
        y_calc[i] = c + d * x[i];
    }
    SAMPLE;
}
```

```

int main(){
    float x[1000];
    float y[1000];
    float y_calc[1000];
    float a, b;
    int i;
    for(i = 0; i < 1000; i++){
        x[i] = i / 3.0;
        y[i] = i * i / 3.0;
    }
    linfit2(x, y, y_calc, &a, &b, 1000);
}

```