**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. 8303 |  | Гришин К. И. |
| Преподаватель |  | Кирьянчиков В. А. |

Санкт-Петербург

2021

# Цель работы

Изучить возможности измерения динамических характеристик программ с помощью профилировщика на примере планировщика Sampler.

# Задание

Разбить программу из первой лабораторной на функциональные участки, определить узкое место и внести изменения для повешения производительности.

# Ход выполнения

## Тестовые программы

Результаты запуска программы test\_cyc.cpp и test\_sub.cpp представлены в таблицах 1 и 2 соотв. Профилирование программ проводилось с использованием параметров: 100 запусков, 10 пропусков без оптимизаций компилятора.

| исх | прием | общее время | кол-во проходов | среднее время |
| --- | --- | --- | --- | --- |
| 13 | 15 | 3793.200 | 1 | 3793.200 |
| 15 | 17 | 7459.100 | 1 | 7459.100 |
| 17 | 19 | 19624.800 | 1 | 19624.800 |
| 19 | 21 | 37528.700 | 1 | 37528.700 |
| 21 | 24 | 3963.500 | 1 | 3963.500 |
| 24 | 27 | 7777.000 | 1 | 7777.000 |
| 27 | 30 | 18443.600 | 1 | 18443.600 |
| 30 | 33 | 37498.100 | 1 | 37498.100 |
| 33 | 39 | 3679.800 | 1 | 3679.800 |
| 39 | 45 | 7540.000 | 1 | 7540.000 |
| 45 | 51 | 18709.200 | 1 | 18709.200 |
| 51 | 57 | 36343.700 | 1 | 36343.700 |

Таблица 1.Результат профилирования test\_cyc.cpp

| исх | прием | общее время | кол-во проходов | среднее время |
| --- | --- | --- | --- | --- |
| 30 | 32 | 30496332.300 | 1 | 30496332.300 |
| 32 | 34 | 61231491.300 | 1 | 61231491.300 |
| 34 | 36 | 153969222.100 | 1 | 153969222.100 |
| 36 | 38 | 308683976.700 | 1 | 308683976.700 |

Таблица 2.Результат профилирования test\_sub.cpp

Из таблиц 1 и 2 видно, что с увеличением количества итераций цикла, увеличивается время работы функциональных участков. Причем можно заметить, что что изменение количества итераций линейно влияет на время исполнения функционального участка.

Из таблицы 1 видно, что массивы одинакового размера обрабатываются за одинаковое время, что говорит о том, что кеширования не происходило.

## Программа из первой лабораторной работы

Для профилирования программы в нее были добавлены контрольные точки для разбиения на функциональные участки (прил. А).

Результат профилирования программы представлен в таблице 3.

| исх | прием | общее время | кол-во проходов | среднее время |
| --- | --- | --- | --- | --- |
| 14 | 18 | 11.556 | 1 | 11.556 |
| 18 | 21 | 25.331 | 1 | 25.331 |
| 21 | 23 | 178.043 | 1 | 178.043 |
| 23 | 26 | 1.164 | 1 | 1.164 |
| 26 | 28 | 1.749 | 1 | 1.749 |
| 28 | 32 | 22.651 | 11 | 2.059 |
| 32 | 35 | 49.813 | 11 | 4.528 |
| 35 | 38 | 35330.592 | 2047 | 17.260 |
| 38 | 40 | 94.427 | 11 | 8.584 |
| 38 | 35 | 3542.309 | 2036 | 1.740 |
| 40 | 42 | 26.520 | 11 | 2.411 |
| 42 | 28 | 84.406 | 10 | 8.441 |
| 42 | 44 | 3.151 | 1 | 3.151 |

Таблица 3. Результат профилирования программы из первой лабораторной работы.

Из таблицы видно, что наибольшее время занимают строки 21-23, однако в данных строках время обусловлено интегрируемой функцией и является частью пользовательского ввода.

В таком случае необходимо произвести оптимизацию остальных частей программы.

Было произведено улучшение программы:

* Теперь функция не принимает указатель на переменную, в которой хранится результат, а возвращает его.
* Все операции для разыменования и взятия адреса устранены.
* Объявление и определение переменных объединены.

Исходный код модифицированной программы представлен в прил. Б.

Результат профилирования программы представлен в таблице 4.

| исх | прием | общее время | кол-во проходов | среднее время |
| --- | --- | --- | --- | --- |
| 14 | 19 | 10.416 | 1 | 10.416 |
| 19 | 22 | 12.880 | 1 | 12.880 |
| 22 | 24 | 103.872 | 1 | 103.872 |
| 24 | 28 | 0.740 | 1 | 0.740 |
| 28 | 38 | 0.857 | 1 | 0.857 |
| 38 | 42 | 14.292 | 11 | 1.299 |
| 42 | 45 | 31.388 | 11 | 2.853 |
| 45 | 48 | 31594.158 | 2047 | 15.434 |
| 48 | 50 | 85.737 | 11 | 7.794 |
| 48 | 45 | 2801.228 | 2036 | 1.376 |
| 50 | 52 | 13.254 | 11 | 1.205 |
| 52 | 38 | 48.192 | 10 | 4.819 |
| 52 | 54 | 4.892 | 1 | 4.892 |

Таблица 4. Результат профилирования модифицированной программы из первой лабораторной.

Из таблицы видно, что большая часть участков кода занимает значительно меньшее время, вплоть до 50%.

Время исполнения программы уменьшилось с 39371.712 до 34721.906, т. е. на 4649.806.

# Вывод

В ходе выполнения лабораторной работы изучены возможности измерения динамических характеристик программ с помощью профилировщика на примере планировщика Sampler.

Проведено профилирование двух тестовых программ, в результате которого видно, как изменяется оценка профилировщика в зависимости от количества итераций.

Проведено профилирование программы из первой лабораторной работы, в которой были обнаружены и устранены узкие места.

Программа была улучшена и при повторном измерении были измерены лучшие временные характеристики.

# Приложение А

1. #include "sampler.h"
2. #include <stdio.h>
3. #include <math.h>
5. const double tol = 1.0e-6;
6. double sum, upper, lower;
8. double fx(double x)
9. {
10. return 1.0 / sqrt(x);
11. }
13. void trapez(double lower, double upper, double tol, double\* sum) {
14. SAMPLE;
15. int pieces, i;
16. double x, delta\_x, end\_sum, mid\_sum, sum1;
18. SAMPLE;
19. pieces = 1;
20. delta\_x = (upper - lower)/pieces;
21. SAMPLE;
22. end\_sum = fx(lower) + fx(upper);
23. SAMPLE;
24. \*sum = end\_sum \* delta\_x / 2.0;
25. mid\_sum = 0.0;
26. SAMPLE;
27. do {
28. SAMPLE;
29. pieces = pieces\*2;
30. sum1 = \*sum;
31. delta\_x = (upper-lower)/pieces;
32. SAMPLE;
33. for(i = 1; i <= pieces / 2; i++)
34. {
35. SAMPLE;
36. x = lower + delta\_x \* (2.0 \* i - 1.0);
37. mid\_sum = mid\_sum + fx(x);
38. SAMPLE;
39. }
40. SAMPLE;
41. \*sum = (end\_sum + 2.0 \* mid\_sum) \* delta\_x \* 0.5;
42. SAMPLE;
43. } while ( fabs(\*sum - sum1) > fabs(tol\*(\*sum)));
44. SAMPLE;
45. }
47. int main(int argc, char \*\*argv) {
48. sampler\_init(&argc, argv);
49. lower = 1.0;
50. upper = 9.0;
51. // printf("\n");
52. trapez(lower,upper,tol,&sum);
53. // printf("\n");
54. // printf("area= %.16e\n", sum);
55. return 0;
56. }

# Приложение Б

1. #include "sampler.h"
2. #include <stdio.h>
3. #include <math.h>
5. const double tol = 1.0e-6;
6. double upper, lower;
8. double fx(double x)
9. {
10. return 1.0 / sqrt(x);
11. }
13. double trapez(double lower, double upper, double tol) {
14. SAMPLE;
15. double x;
16. double sum1;
17. int i;
19. SAMPLE;
20. int pieces = 1;
21. double delta\_x = (upper - lower)/pieces;
22. SAMPLE;
23. double end\_sum = fx(lower) + fx(upper);
24. SAMPLE;
26. double mid\_sum = 0.0;
27. double sum = end\_sum \* delta\_x / 2.0;
28. SAMPLE;
29. do {
30. SAMPLE;
31. pieces = pieces << 1;
32. sum1 = sum;
33. delta\_x = (upper-lower)/pieces;
34. SAMPLE;
35. for(i = 1; i <= pieces >> 1; i++)
36. {
37. SAMPLE;
38. x = lower + delta\_x \* (2.0 \* i - 1.0);
39. mid\_sum = mid\_sum + fx(x);
40. SAMPLE;
41. }
42. SAMPLE;
43. sum = (end\_sum + 2.0 \* mid\_sum) \* delta\_x \* 0.5;
44. SAMPLE;
45. } while (fabs(sum - sum1) > fabs(tol\*sum));
46. SAMPLE;
47. return sum;
48. }
50. int main(int argc, char \*\*argv) {
51. sampler\_init(&argc, argv);
52. lower = 1.0;
53. upper = 9.0;
54. // printf("\n");
55. double sum = trapez(lower,upper,tol);
56. // printf("\n");
57. // printf("area= %.16e\n", sum);
58. return 0;
59. }