**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. 7304 |  | Субботин А.С. |
| Преподаватель |  | Ефремов М.А. |

Санкт-Петербург

2021

**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:

a. измерение только полного времени выполнения программы;

b. 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.

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

**Исходные данные**

Вариант 17. Исходный код для пунктов 2-3 задания был получен преобразованием исходного кода на Си из лабораторной работы №1 соответствующего варианта.

**Ход работы**

Для использования была выбрана старая версия монитора Sampler\_old, соответственно запуск производился под эмулятором DOSBox-0.74

1. Произведено профилирование тестового файла TEST\_CYC.EXE. Исходный код программы представлен в Приложении А. Результаты профилирования представлены на Рисунке 1.

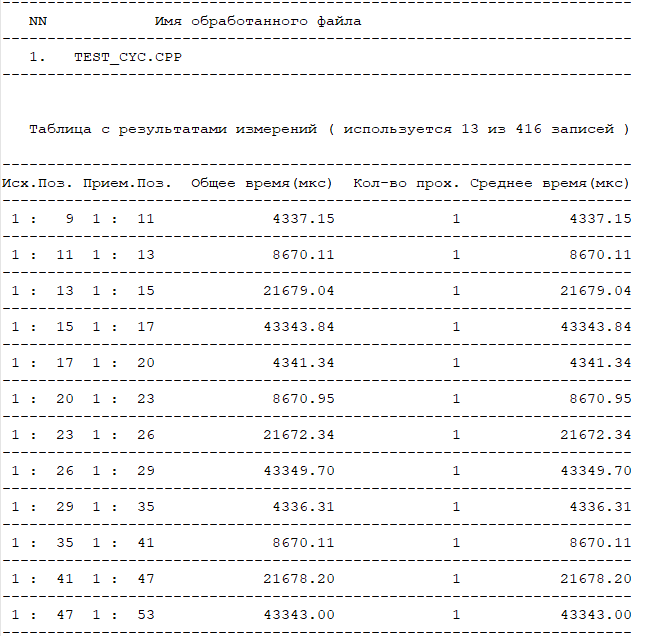


Рисунок 1 – Результат профилирования TEST\_CYC.EXE

Участок 9-11 в 10 раз меньше, 11-13 в 5 раз меньше, 13-15 в 2 раза меньше участка 15-17, что согласуется с соответственной кратной разницей в количестве итераций циклов.

1. Произведено профилирование тестового файла TEST\_SUB.EXE. Исходный код программы представлен в Приложении Б. Результаты профилирования представлены на Рисунке 2.

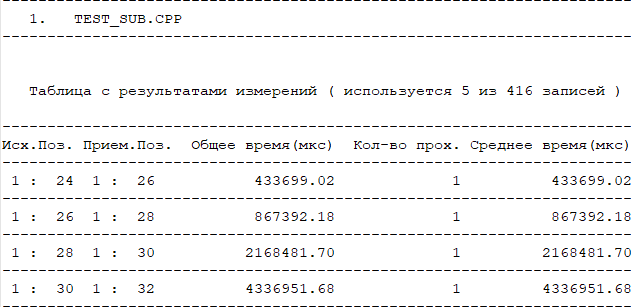


Рисунок 2 – Результат профилирования TEST\_SUB.EXE

Как и в пункте 1, затраченное время на выполнение функции уменьшается столько же кратно, сколь кратно уменьшается количество вызовов этой функции (в 2, в 5 и в 10 раз).

1. Был взят код, написанный в ходе выполнения ЛР №1. Для того, чтобы команда компилировалась, была произведена замена «#include <stdbool.h>» на «typedef enum { false, true } bool;», т.к. изначальный файл был добавлен в более поздние версии языка Си.
2. Полный замер (код в Приложении В) представлен на Рисунке 3.

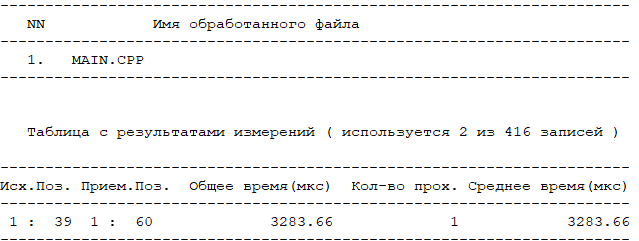


Рисунок 3 – Измерение полного времени выполнения программы

1. Отдельные замеры (код в Приложении Г) представлен на Рисунке 4.

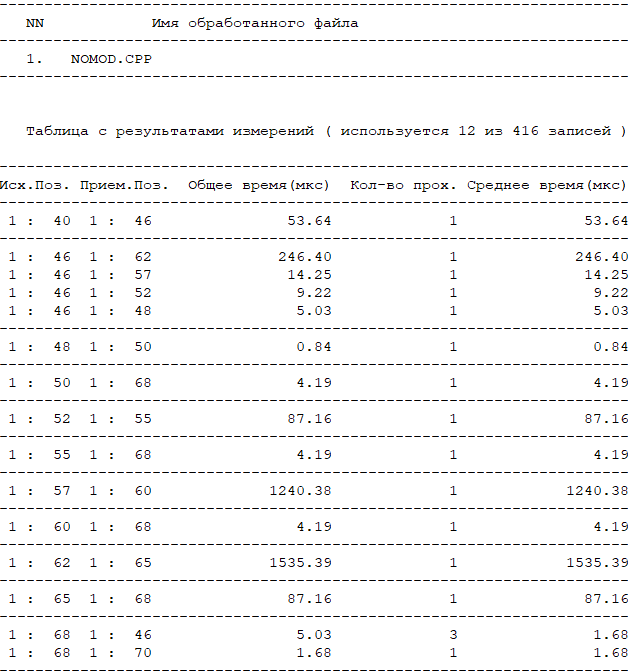


Рисунок 4 – Измерение времен выполнения ФУ

53.64 + 246.4 + 14.25 + 9.22 + 5.03 + 0.84 + 4.19 + 87.16 + 4.19 + 1240.38 + 4.19 +1535.39 + 87.16 + 5.03 + 1.68 = 3298.75 – Суммарное время отдельных замеров.

Время выполнения программы с отдельными замерами (3298.75 мкс) больше полного (3283.66 мкс) на 15 мкс (0,5%).

1. Из-за невозможности улучшения реализации вычислительных функций возможно лишь незначительное улучшение производительности. Изменения:

* Избавление от переменной «done»
* Замена цикла do while на for
* (потенциально) Если бы Turbo C++ 3.0 поддерживал стандарт С99, то сами функции ошибок можно было бы отдать библиотеке math.h, а не считать разложением в ряд

Модифицированный код представлен в приложении Д.

1. Измерение полного времени выполнения модифицированной программы представлено на Рисунке 5.

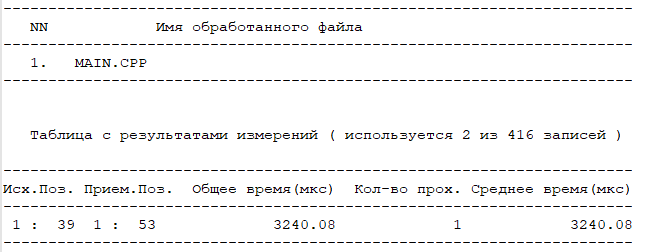


Рисунок 5 – Измерение полного времени выполнения модифицированной программы

С модифицированным кодом удалось выиграть около 43 мкс.

**Выводы**

В ходе выполнения данной лабораторной работы было произведено вычисление профиля программы на Си с помощью профилировщика Sampler. Произведена оптимизация программы из ЛР1.

**Приложение А. TEST\_CYC.CPP**

1 #include <stdlib.h>

2 #include “Sampler.h”

3 #define Size 10000

4

5 int i, tmp, dim[Size];

6

7 void main()

8 {

9 SAMPLE;

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

11 SAMPLE;

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

13 SAMPLE;

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

15 SAMPLE;

16 for(i=0;i<Size;i++) { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };

17 SAMPLE;

18 for(i=0;i<Size/10;i++)

19 { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };

20 SAMPLE;

21 for(i=0;i<Size/5;i++)

22 { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };

23 SAMPLE;

24 for(i=0;i<Size/2;i++)

25 { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };

26 SAMPLE;

27 for(i=0;i<Size;i++)

28 { tmp=dim[0]; dim[0]=dim[i]; dim[i]=tmp; };

29 SAMPLE;

30 for(i=0;i<Size/10;i++)

31 { tmp=dim[0];

32 dim[0]=dim[i];

33 dim[i]=tmp;

34 };

35 SAMPLE;

36 for(i=0;i<Size/5;i++)

37 { tmp=dim[0];

38 dim[0]=dim[i];

39 dim[i]=tmp;

40 };

41 SAMPLE;

42 for(i=0;i<Size/2;i++)

43 { tmp=dim[0];

44 dim[0]=dim[i];

45 dim[i]=tmp;

46 };

47 SAMPLE;

48 for(i=0;i<Size;i++)

49 { tmp=dim[0];

50 dim[0]=dim[i];

51 dim[i]=tmp;

52 };

53 SAMPLE;

54 }

**Приложение Б. TEST\_SUB.CPP**

1 #include <stdlib.h>

2 #include “Sampler.h”

3 const unsigned Size = 1000;

4 void TestLoop(int nTimes)

5 {

6 static int TestDim[Size];

7 int tmp;

8 int iLoop;

9 while (nTimes > 0)

10 {

11 nTimes --;

12 iLoop = Size;

13 while (iLoop > 0)

14 {

15 iLoop -- ;

16 tmp = TestDim[0];

17 TestDim[0] = TestDim[nTimes];

18 TestDim[nTimes] = tmp;

19 }

20 }

21 } /\* TestLoop \*/

22 void main()

23 {

24 SAMPLE;

25 TestLoop(Size / 10); // 100 \* 1000 повторений

26 SAMPLE;

27 TestLoop(Size / 5); // 200 \* 1000 повторений

28 SAMPLE;

29 TestLoop(Size / 2); // 500 \* 1000 повторений

30 SAMPLE;

31 TestLoop(Size / 1); // 1000\* 1000 повторений

32 SAMPLE;

33 }

**Приложение В. Код программы из ЛР №1 для проведения замера полного времени выполнения программы**

1. #include <stdio.h>

2. #include <stdlib.h>

3. #include <math.h>

4. #include "sampler.h"

5. typedef enum {false, true} bool;

6. double erf(double x){

7. const double sqrtpi = 1.7724538;

8. double t2 = 0.66666667;

9. double t3 = 0.66666667;

10. double t4 = 0.07619048;

11. double t5 = 0.01693122;

12. double t6 = 3.078403E-3;

13. double t7 = 4.736005E-4;

14. double t8 = 6.314673E-5;

15. double t9 = 7.429027E-6;

16. double t10 = 7.820028E-7;

17. double t11 = 7.447646E-8;

18. double t12 = 6.476214E-9;

19.

20. double x2, sum;

21. x2 = x\*x;

22. sum = t5+x2\*(t6+x2\*(t7+x2\*(t8+x2\*(t9+x2\*(t10+x2\*(t11+x2\*t12))))));

23. return (2.0\*exp(-x2)/sqrtpi\*(x\*(1+x2\*(t2+x2\*(t3+x2\*(t4+x2\*sum))))));

24.

25. }

26.

27. double erfc(double x){

28. const double sqrtpi = 1.7724538;

29. double x2,v,sum;

30. x2 = x\*x;

31. v = 1/(2\*x2);

32. sum=v/(1+8\*v/(1+9\*v/(1+10\*v/(1+11\*v/(1+12\*v)))));

33. sum=v/(1+3\*v/(1+4\*v/(1+5\*v/(1+6\*v/(1+7\*sum)))));

34. return (1.0/(exp(x2)\*x\*sqrtpi\*(1+v/(1+2\*sum))));

35. }

36.

37. int main()

38. {

39. SAMPLE;

40. double x,er,ec;

41. bool done;

42. x = 2.0;

43. done = false;

44. do{

45. if(x<0){

46. done = true;

47. }else if (x == 0){

48. er = 0;

49. ec = 1;

50. }else if (x < 1.5){

51. er = erf(x);

52. ec = 1 - er;

53. }else{

54. ec = erfc(x);

55. er = 1-ec;

56. }

57. x = x - 1;

58.

59. }while (done == false);

60. SAMPLE;

61. return 0;

62. }

**Приложение Г. Код программы из ЛР №1 для измерения времен выполнения ФУ**

1. #include <stdio.h>

2. #include <stdlib.h>

3. #include <math.h>

4. #include "SAMPLER.H"

5. typedef enum { false, true } bool;

6.

7. double erf(double x){

8. const double sqrtpi = 1.7724538;

9. double t2 = 0.66666667;

10. double t3 = 0.66666667;

11. double t4 = 0.07619048;

12. double t5 = 0.01693122;

13. double t6 = 3.078403E-3;

14. double t7 = 4.736005E-4;

15. double t8 = 6.314673E-5;

16. double t9 = 7.429027E-6;

17. double t10 = 7.820028E-7;

18. double t11 = 7.447646E-8;

19. double t12 = 6.476214E-9;

20.

21. double x2, sum;

22. x2 = x\*x;

23. sum = t5+x2\*(t6+x2\*(t7+x2\*(t8+x2\*(t9+x2\*(t10+x2\*(t11+x2\*t12))))));

24. return (2.0\*exp(-x2)/sqrtpi\*(x\*(1+x2\*(t2+x2\*(t3+x2\*(t4+x2\*sum))))));

25.

26. }

27.

28. double erfc(double x){

29. const double sqrtpi = 1.7724538;

30. double x2,v,sum;

31. x2 = x\*x;

32. v = 1/(2\*x2);

33. sum=v/(1+8\*v/(1+9\*v/(1+10\*v/(1+11\*v/(1+12\*v)))));

34. sum=v/(1+3\*v/(1+4\*v/(1+5\*v/(1+6\*v/(1+7\*sum)))));

35. return (1.0/(exp(x2)\*x\*sqrtpi\*(1+v/(1+2\*sum))));

36. }

37.

38. int main()

39. {

40. SAMPLE;

41. double x,er,ec;

42. bool done;

43. x = 2.0;

44. done = false;

45. do{

46. SAMPLE;

47. if(x<0){

48. SAMPLE;

49. done = true;

50. SAMPLE;

51. }else if (x == 0){

52. SAMPLE;

53. er = 0;

54. ec = 1;

55. SAMPLE;

56. }else if (x < 1.5){

57. SAMPLE;

58. er = erf(x);

59. ec = 1 - er;

60. SAMPLE;

61. }else{

62. SAMPLE;

63. ec = erfc(x);

64. er = 1-ec;

65. SAMPLE;

66. }

67. x = x - 1;

68. SAMPLE;

69. }while (done == false);

70. SAMPLE;

71. return 0;

72. }

**Приложение Д. Модифицированный код программы из ЛР №1 для проведения замера полного времени выполнения программы**

1. #include <stdio.h>

2. #include <stdlib.h>

3. #include <math.h>

4. #include "SAMPLER.H"

5.

6. double erf(double x){

7. const double sqrtpi = 1.7724538;

8. double t2 = 0.66666667;

9. double t3 = 0.66666667;

10. double t4 = 0.07619048;

11. double t5 = 0.01693122;

12. double t6 = 3.078403E-3;

13. double t7 = 4.736005E-4;

14. double t8 = 6.314673E-5;

15. double t9 = 7.429027E-6;

16. double t10 = 7.820028E-7;

17. double t11 = 7.447646E-8;

18. double t12 = 6.476214E-9;

19.

20. double x2, sum;

21. x2 = x\*x;

22. sum = t5+x2\*(t6+x2\*(t7+x2\*(t8+x2\*(t9+x2\*(t10+x2\*(t11+x2\*t12))))));

23. return (2.0\*exp(-x2)/sqrtpi\*(x\*(1+x2\*(t2+x2\*(t3+x2\*(t4+x2\*sum))))));

24.

25. }

26.

27. double erfc(double x){

28. const double sqrtpi = 1.7724538;

29. double x2,v,sum;

30. x2 = x\*x;

31. v = 1/(2\*x2);

32. sum=v/(1+8\*v/(1+9\*v/(1+10\*v/(1+11\*v/(1+12\*v)))));

33. sum=v/(1+3\*v/(1+4\*v/(1+5\*v/(1+6\*v/(1+7\*sum)))));

34. return (1.0/(exp(x2)\*x\*sqrtpi\*(1+v/(1+2\*sum))));

35. }

36.

37. int main()

38. {

39. SAMPLE;

40. double x,er,ec;

41. for(x = 0; x<=2.0; x++){

42. if (x == 0){

43. er = 0;

44. ec = 1;

45. }else if (x < 1.5){

46. er = erf(x);

47. ec = 1 - er;

48. }else{

49. ec = erfc(x);

50. er = 1-ec;

51. }

52. }

53. SAMPLE;

54. return 0;

55. }