

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

**ОТЧЕТ**

**по практической работе №3**

**по дисциплине «Качество и метрология программного обеспечения»**

**ТЕМА: ИЗМЕРЕНИЕ ХАРАКТЕРИСТИК ДИНАМИЧЕСКОЙ СЛОЖНОСТИ**

**ПРОГРАММ С ПОМОЩЬЮ ПРОФИЛИРОВЩИКА SAMPLER\_v2**

Студент гр. 8304

Преподаватель

Нам Ё Себ

Ефремов М.А.

Санкт-Петербург

2022


## Цель работы.

Изучить возможности измерения динамических характеристик программ с помощью профилировщика на примере профилировщика SAMPLER

## Ход работы.

1. Были выполнены тестовые программы test\_cyc.c и test\_sub.c под управлением SAMPLER

Таблица 1 – Результаты для test\_cyc.c



исх	прием	общее время	кол-во проходов	среднее время
13	15	5810.000	1	5810.000
15	17	9190.000	1	9190.000
17	19	23010.000	1	23010.000
19	21	38340.000	1	38340.000
21	24	2890.000	1	2890.000
24	27	5690.000	1	5690.000
27	30	14210.000	1	14210.000
30	33	28260.000	1	28260.000
33	39	2890.000	1	2890.000
39	45	5670.000	1	5670.000
45	51	14150.000	1	14150.000
51	57	28250.000	1	28250.000






Таблица 2 – Результаты для test\_sub.c



исх	прием	общее время	кол-во проходов	среднее время
30	32	25913830.000	1	25913830.000
32	34	51927060.000	1	51927060.000
34	36	131363580.000	1	131363580.000
36	38	262845350.000	1	262845350.000



2. Выполнили программу из ЛР1 под управлением Sampler с внешним заикливанием и получили отчет по результатам профилирования.

<b>исх</b>	<b>прием</b>	<b>общее время</b>	<b>кол-во проходов</b>	<b>среднее время</b>
21	26	13.042	1	13.042
26	28	49.526	1	49.526
28	33	77.258	5	15.452
28	31	115.970	6	19.328
33	38	65.342	5	13.068
38	40	11.833	1	11.833
38	54	77.712	4	19.428
40	45	42.919	1	42.919
45	48	39.878	1	39.878
48	54	10.861	1	10.861
54	56	54.001	5	10.800
56	59	80.530	6	13.422
59	62	65.440	6	10.907
59	59	39.877	2	19.939
62	65	58.482	3	19.494
62	67	37.047	3	12.349
65	65	30.821	3	10.274
65	67	73.477	3	24.492
67	70	22.324	1	22.324
67	73	63.570	5	12.714
70	56	8.232	1	8.232
73	75	72.569	5	14.514
75	77	134.443	5	26.889
77	87	58.017	2	29.008
77	82	29.819	3	9.940
87	90	15.648	2	7.824
90	28	27.947	5	5.589
82	90	24.116	3	8.039
31	28	217.023	5	43.405
31	93	29.219	1	29.219

Как видно из результатов измерения времени выполнения функциональных участков – наиболее затратным по среднему времени фрагментом является пропуск тела else и переход с 31 строки на 28. Для оптимизации добавим оператор «continue», чтобы сразу переходить к проверке условия цикла.

Для уменьшения времени прохождения кода 28-31 и 28-33 операции взятия элемента массива по индексу в условии if были заменены адресной арифметикой. Так же для 38-54 в строке 53 произведена замена на адресную арифметику.

Для 56-59, 65-67, 73-75 так же заменены операции взятия элемента массива по индексу.

Была выполнена проверка изменённой программы. Результат представлен на рисунке 5. Исходный код модифицированной программы представлен в Приложении В.

<b>исх</b>	<b>прием</b>	<b>общее время</b>	<b>кол-во проходов</b>	<b>среднее время</b>
21	26	15.734	1	15.734
26	28	49.641	1	49.641
28	34	61.912	5	12.382
28	31	3.745	6	0.624
34	39	72.786	5	14.557
39	41	11.348	1	11.348
39	55	59.045	4	14.761
41	46	42.453	1	42.453
46	49	34.411	1	34.411
49	55	12.173	1	12.173
55	57	77.268	5	15.454
57	60	70.817	6	11.803
60	63	28.873	6	4.812
60	60	42.626	2	21.313
63	66	38.050	3	12.683
63	68	22.359	3	7.453
66	66	30.986	3	10.328
66	68	55.556	3	18.519
68	71	25.861	1	25.861
68	74	51.304	5	10.261
71	57	7.832	1	7.832
74	76	42.281	5	8.456
76	78	109.711	5	21.942
78	88	45.866	2	22.933
78	83	30.836	3	10.279
88	91	5.626	2	2.813
91	28	3.817	5	0.763
83	91	28.672	3	9.557
31	28	53.113	5	10.623
31	94	23.101	1	23.101

Из общего времени выполнения в 1646,94 получилось 1157,80

### **Заключение**

В ходе лабораторной работы изучили возможности измерения динамических характеристик программ с помощью профилировщика на примере профилировщика SAMPLER.

## Приложение А. Исходный код программы

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "sampler.h"

#define max 10

void swap(double* p, double* q)
{
    double hold = *p;
    *p = *q;
    *q = hold;
}

void sort(double* x, int n)
{
    int left[20];
    int right[20];
    int i, j, sp, mid;
    double pivot;

    left[0] = 0;
    right[0] = n-1;
    sp = 0;

    while (sp > -1){
        if (left[sp] >= right[sp]) {
            sp--;
        } else {
            i = left[sp];
            j = right[sp];
            pivot = x[j];
            mid = (i + j) / 2; //div
```

```

        if ((j - i) > 5){
            if (((x[mid] < pivot) && (x[mid] > x[i])) || ((x[mid] > pivot)
&& (x[mid] < x[i]))){
                swap(x + mid, x + j);
            } else {
                if (((x[i] < x[mid]) && (x[i] > pivot)) || ((x[i] >
x[mid]) && (x[i] < pivot)))
                    swap(x + i, x + j);
            }
        }
    }

    pivot = x[j];
    while (i < j) {
        while (x[i] < pivot)
            i++;
        j = j - 1;
        while ((i < j) && (pivot < x[j]))
            j--;
        if (i < j)
            swap(x + i, x + j);
    }

    j = right[sp];
    swap(x + i, x + j);

    if (i - left[sp] >= right[sp] - i){
        left[sp + 1] = left[sp];
        right[sp + 1] = i - 1;
        left[sp] = i + 1;
    } else {
        left[sp + 1] = i + 1;
        right[sp + 1] = right[sp];
        right[sp] = i - 1;
    }
    sp++;
}
}

```

```
}
```

```
int main(int argc, char **argv)
```

```
{
```

```
    sampler_init(&argc, argv);
```

```
    int n = max;
```

```
    double x[max] = {2.1, 3.0, 0.3, 5.6, 2.7, 1.0, 45.9, 10.0, 94.2, 0.01};
```

```
    SAMPLE;
```

```
    sort(x, n);
```

```
    SAMPLE;
```

```
    return 0;
```

```
}
```

## Приложение Б. Исходный код программы с разделением на ФУ

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "sampler.h"

#define max 10

void swap(double* p, double* q)
{
    double hold = *p;
    *p = *q;
    *q = hold;
}

void sort(double* x, int n)
{
    int left[20];
    int right[20];
    int i, j, sp, mid;
    double pivot;
    SAMPLE;
    left[0] = 0;
    right[0] = n-1;
    sp = 0;

    SAMPLE;
    while (sp > -1){
        SAMPLE;
        if (left[sp] >= right[sp]) {
            sp--;
            SAMPLE;
        } else {
            SAMPLE;
            i = left[sp];
```



```

    j = right[sp];
    pivot = x[j];
    mid = (i + j) / 2; //div
    SAMPLE;
    if ((j - i) > 5){
        SAMPLE;
        if (((x[mid] < pivot) && (x[mid] > x[i])) || ((x[mid] > pivot)
&& (x[mid] < x[i]))){
            swap(x + mid, x + j);
            SAMPLE;
        } else {
            SAMPLE;
            if (((x[i] < x[mid]) && (x[i] > pivot)) || ((x[i] >
x[mid]) && (x[i] < pivot))){
                swap(x + i, x + j);
                SAMPLE;
            }
        }
    }

    pivot = x[j];
    SAMPLE;
    while (i < j){
        SAMPLE;
        while (x[i] < pivot){
            i++;
            SAMPLE;
        }
        j = j - 1;
        SAMPLE;
        while ((i < j) && (pivot < x[j])){
            j--;
            SAMPLE;
        }
        SAMPLE;
        if (i < j){
            swap(x + i, x + j);

```

```

        SAMPLE;
    }
}
SAMPLE;
j = right[sp];
SAMPLE;
swap(x + i, x + j);
SAMPLE;
if (i - left[sp] >= right[sp] - i){
    left[sp + 1] = left[sp];
    right[sp + 1] = i - 1;
    left[sp] = i + 1;
    SAMPLE;
} else {
    left[sp + 1] = i + 1;
    right[sp + 1] = right[sp];
    right[sp] = i - 1;
    SAMPLE;
}
sp++;
SAMPLE;
}
}
SAMPLE;
}

int main(int argc, char **argv)
{
    sampler_init(&argc, argv);

    int n = max;
    double x[max] = {2.1, 3.0, 0.3, 5.6, 2.7, 1.0, 45.9, 10.0, 94.2, 0.01};
    sort(x, n);
    return 0;
}

```



## Приложение В. Исходный код программы с оптимизациями

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "sampler.h"

#define max 10

void swap(double* p, double* q)
{
    double hold = *p;
    *p = *q;
    *q = hold;
}

void sort(double* x, int n)
{
    int left[20];
    int right[20];
    int i, j, sp, mid;
    double pivot;
    SAMPLE;
    left[0] = 0;
    right[0] = n-1;
    sp = 0;

    SAMPLE;
    while (sp > -1){
        SAMPLE;
        if (*(left+sp) >= *(right+sp)) {
            sp--;
            SAMPLE;
            continue;
        } else {
            SAMPLE;
            i = left[sp];
            j = right[sp];
            pivot = x[j];
            mid = (i + j) / 2; //div
            SAMPLE;
            if ((j - i) > 5){
                SAMPLE;
                if (((x[mid] < pivot) && (x[mid] > x[i])) || ((x[mid] > pivot)
&& (x[mid] < x[i]))){
                    swap(x + mid, x + j);
                    SAMPLE;
                } else {
                    SAMPLE;
                    if (((x[i] < x[mid]) && (x[i] > pivot)) || ((x[i] >
x[mid]) && (x[i] < pivot)))){
                        swap(x + i, x + j);
                        SAMPLE;
                    }
                }
            }
        }
    }
}
```

```

        pivot = *(x+j);
        SAMPLE;
        while (i < j){
            SAMPLE;
            while (*(x+i) < pivot){
                i++;
                SAMPLE;
            }
            j = j - 1;
            SAMPLE;
            while ((i < j) && (pivot < *(x+j))){
                j--;
                SAMPLE;
            }
            SAMPLE;
            if (i < j){
                swap(x + i, x + j);
                SAMPLE;
            }
        }
        SAMPLE;
        j = *(right + sp);
        SAMPLE;
        swap(x + i, x + j);
        SAMPLE;
        if (i - left[sp] >= right[sp] - i){
            left[sp + 1] = left[sp];
            right[sp + 1] = i - 1;
            left[sp] = i + 1;
            SAMPLE;
        } else {
            left[sp + 1] = i + 1;
            right[sp + 1] = right[sp];
            right[sp] = i - 1;
            SAMPLE;
        }
        sp++;
        SAMPLE;
    }
}
SAMPLE;
}

int main(int argc, char **argv)
{
    sampler_init(&argc, argv);

    int n = max;
    double x[max] = {2.1, 3.0, 0.3, 5.6, 2.7, 1.0, 45.9, 10.0, 94.2, 0.01};
    sort(x, n);
    return 0;
}

```