Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.01 Информатика и вычислительная техника**

ОТЧЕТ

по лабораторной работе № 3

Вариант 11

Название: Оценка эффективности и качества программы

Дисциплина: Технологии разработки программных продуктов

Студент ИУ6-43Б (Группа) _____ (Подпись, дата) В.К. Залыгин (И.О. Фамилия)

Преподаватель _____ (Подпись, дата) Е.К. Пугачев (И.О. Фамилия)

Москва, 2024

1 Цель лабораторной работы

Цель данной работы — изучить известные критерии оценки и способы повышения эффективности и качества программных продуктов.

2 Описание задания

Задание представлено на рисунке 1.

11. Написать программу вычисления суммы ряда $S = \frac{1}{4} - \frac{1}{16} + \frac{1}{96} - \ldots$ $\ldots (-1)^{n+1} \frac{1}{4^n \cdot n!}$ с точностью 0,0001. Ряд сходится и имеет множитель $m = -\frac{1}{4 \cdot n}$ (программа v11.dpr).

Рисунок 1 – Задание

3 Исходный код программы с точками фиксации времен

Исходный код программы представлен в листинге 1.

Листинг 1 – Исходный код программы

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <math.h>

int main()
{
	double s,a,m,eps,x;
	double *ms, *mss;
	int N = 1000, i,o;
	ms = (double*)malloc(N * sizeof(int));
	mss = (double*)malloc(N * sizeof(int));
	eps = 0.00001;
	ms[1] = 1.0/4;
	o = 1;
	s = 1.0/4;
	mss[1] = s;
	do {
	o++;
	m = -1/(4*o);
	ms[o] = ms[o-1]*m;
	s = s+ms[o];
	mss[o] = s;
```

```
        } while (abs(ms[o-1]-ms[o])>=eps);
        printf("s = %f\nn = %d",s,o);
        return 0;
}
```

На листинге 2 представлен исходный код с точками фиксации времени с использованием метода “увеличенной линзы”. Были исправлены ошибки работы программы в строчках 26 (неправильный тип данных) и 30 (вызов неправильной функции).

Листинг 2 – Исправленная программа с “увеличенной линзой” и точками фиксации времени

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <math.h>
#include <time.h>

#include "macros.h"

int main(int argc, char **argv) {
        double s,a,m,eps,x;
        double *ms, *mss;
        int N, i,o;

        START_MEASURE("до оптимизаций\t\t")

        N = 1000;
        ms = (double*)malloc(N * sizeof(int));
        mss = (double*)malloc(N * sizeof(int));
        eps = 0.00001;
        ms[1] = 1.0/4;
        o = 1;
        s = 1.0/4;
        mss[1] = s;
        do {
        o++;
        m = -1.0/(4*o);
        ms[o] = ms[o-1]*m;
        s = s+ms[o];
```

```
        mss[o] = s;
        } while (fabs(ms[o-1]-ms[o])>=eps);

        END_MEASURE()

        printf("s = %f\nn = %d\n",s,o);
        return 0;
}
```

4 Улучшенный вариант программы

Листинг 3 иллюстрирует улучшенную версию программы с точками фиксации времени.

Листинг 3 – Улучшенная программа

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <math.h>
#include <time.h>

#include "macros.h"

int main(int argc, char **argv) {
        double s,m,eps,cur,prev;
        int N,i,o;

        START_MEASURE("после оптимизаций\t")

        N = 1000;
        eps = 0.00001;
        cur = 1.0/4;
        o = 1;
        s = cur;
        do {
        prev = cur;
        o++;
        m = -1.0/(4*o);
        cur = prev*m;
        s = s+cur;
        } while (fabs(prev-cur)>=eps);
```

```
        END_MEASURE()

        printf("s = %f\nn = %d\n",s,o);
        return 0;
}
```

5 Оценка эффективности

Результаты замеров времени представлены на рисунке 2. После улучшений программа работает в 300 раз. Такое ускорение связано с отказом от вызова системной функции malloc, которая значительно замедляет программу, а также удаления обращения по индексу в массивах.

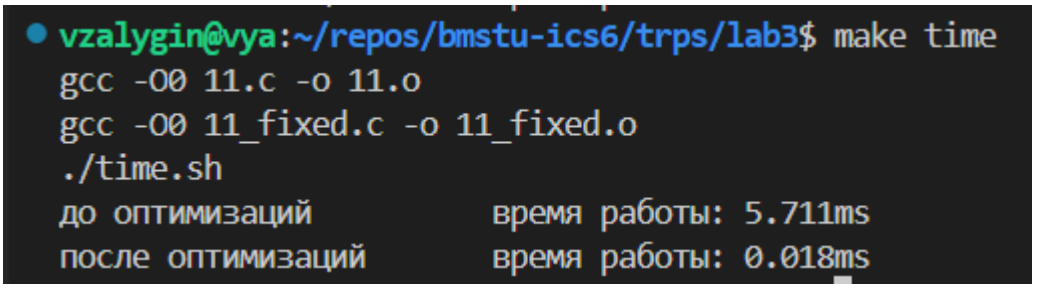

Рисунок 2 – результаты замеров

В таблице 1 отражены результаты замеров времени и оценки памяти для исходной программы и улучшенной, а также указаны недостатки и способы улучшения.

Таблица 1 – Оценка эффективности

Критерии оценки	Исходная программа		Улучшенная программа	
	Недостатки	Количественная оценка	Улучшения	Количественная оценка
Время выполнения	Вызовы тяжеловесной системной функции malloc, повторные вычисления.	5.711мс	Удалены вызовы malloc, повторные операции, обращения к массивам.	0.018мс
Использов	Создаются 2 больших	8068 байт	Удалены массивы,	52 байта

анная память	массива по 1000 элементов, у которых нужно только последние 2 значения		использованы вместо с них переменные	

Оценка качества

В таблице 2 отражены результаты оценки качества исходной программы.

Таблица 2 – оценка качества

Результаты оценки	Критерии оценки			
	Правильность	Универсальность	Проверяемость	Точность результатов
Недостатки	При аллокации указан неправильный размер данных. Вместо sizeof(double) написано sizeof(int).	Программа работает по заранее определенным параметрам, можно обеспечить ввод начальных данных.	Выводится только результат работы и количество итераций. Можно выводить значение для каждой итерации, а также начальные параметры, с которыми работает программа.	–
Оценка, баллы	5	3	3	–

Заключение

В результате проведенных экспериментов были выполнены замеры времени работы программы, оценки памяти, а также предложены способы повышения эффективности и качества программы.