**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет информационных технологий**

**Кафедра параллельных вычислений**

**ОТЧЕТ**

**О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ**

«Определение времени работы прикладной программы»

студента *2* курса, группы 21204

**Осипова Александра Александровича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
к.т.н. Власенко Андрей Юрьевич

Новосибирск 2022

# СОДЕРЖАНИЕ

## ЦЕЛЬ

1. Изучение методики измерения времени работы подпрограммы.
2. Изучение приемов повышения точности измерения времени работы подпрограммы.
3. Изучение способов измерения времени работы подпрограммы.
4. Измерение времени работы подпрограммы в прикладной программе.
5. Изучение основных функций оптимизирующего компилятора, и некоторых примеров оптимизирующих преобразований и уровней оптимизации.
6. Получение базовых навыков работы с компилятором GCC.
7. Исследование влияния оптимизационных настроек компилятора GCC на время исполнения программы.

## ЗАДАНИЕ

1. Написать программу на C или C++, которая реализует алгоритм вычисления числа Пи по формуле Лейбница (1 вариант).

$$\pi = 4\sum_{n=0}^{\infty}\frac{(-1)^n}{2n+1} = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \ldots + 4\frac{(-1)^n}{(2n+1)} + \ldots$$

2. Проверить правильность работы программы на нескольких тестовых набора входных данных.
3. Выбрать значение параметра N таким, чтобы время работы программы было в диапазоне от 30 до 60 секунд.
4. Программу скомпилировать компилятором GCC с уровнями оптимизации -O0, -O1, -O3, -Os, -Ofast, -Og под архитектуру процессора x86.
5. Для каждого из семи вариантов компиляции измерить время работы программы при нескольких значениях N (0.5*N, N, 1.5*N).
6. Составить отчет по лабораторной работе.

# ОПИСАНИЕ РАБОТЫ

1. Была создана папка lab1 на сервере кафедры, куда был добавлен исходный файл lab1.cpp.
2. Была написана функция double PiByGregoryLeibnuz(long long int N), реализующая алгоритм вычисления числа Пи по первым N членам ряда Грегори-Лейбница. (см. Приложение 1)
3. Была проверена правильность работы программы на нескольких тестовых наборов входных данных. (см. Приложение 2)

| Входные данные, N | 0 | 1 | 100 | 1000 | 1000000 |
|---|---|---|---|---|---|
| Число Пи | 4 | 2.666667 | 3.151493 | 3.142592 | 3.141594 |

4. Было выбрано значение N0 таким, чтобы время работы функции было от 30 до 60 секунд. Для исследования была выбрана функция clock_gettime(), получающая значение системного таймера. При N0 = 5 000 000 000 время работы функции составило ~34 секунд. (см. Приложение 3)
5. Программа была скомпилирована компилятором GCC с уровнями оптимизации -O0, -O1, -O2, -O3, -Ofast, -Og (см. Приложение 4).
6. Для каждого из семи вариантов компиляции с помощью утилиты time было измерено общее время работы программы согласно системному таймеру при нескольких значениях N (см. Приложение 5):
    a. N1 = 2 500 000 000
    b. N2 = 5 000 000 000
    c. N3 = 7 500 000 000

| | Базовый уровень | -O0 | -O1 | -O2 | -O3 | -Ofast | -Og | -Os |
|---|---|---|---|---|---|---|---|---|
| N1 | 17.591 | 17.504 | 17.505 | 17.508 | 17.419 | 17.506 | 17.507 | 17.506 |
| N2 | 34.762 | 34.753 | 34.752 | 34.754 | 34.689 | 34.512 | 34.753 | 35.056 |
| N3 | 52.006 | 52.003 | 52.263 | 52.009 | 54.053 | 51.764 | 52.012 | 51.997 |

*Результаты измерений, сек*

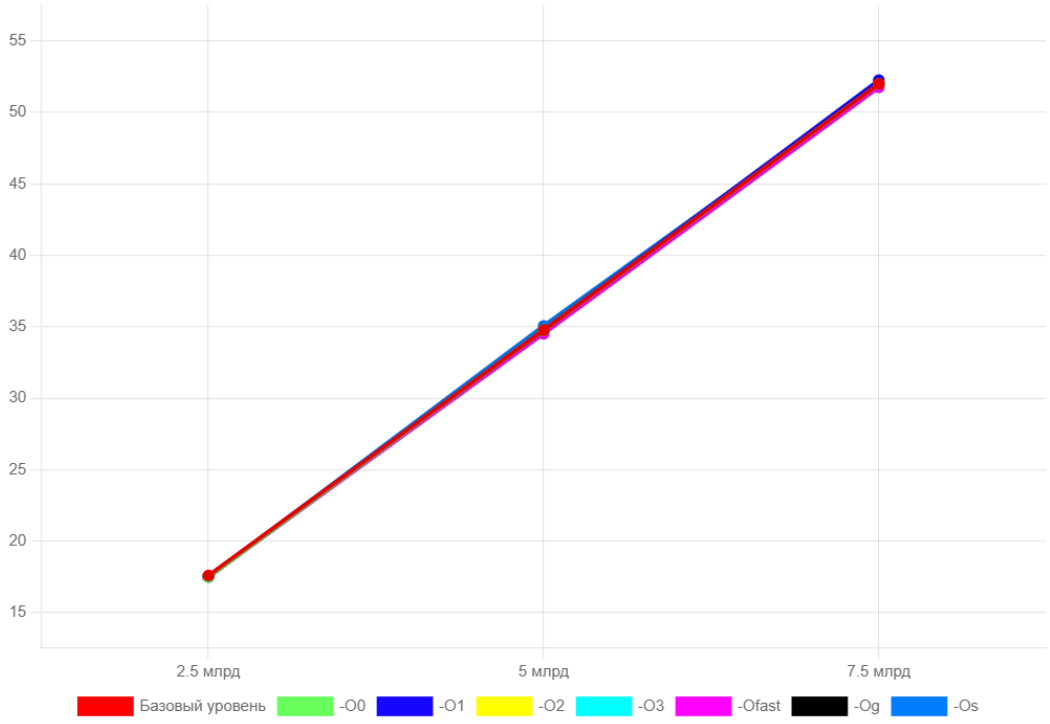


**График результатов измерений**

# ЗАКЛЮЧЕНИЕ

В рамках данной лабораторной работы были изучены такие способы измерения времени работы программы, как утилита time и функция clock_gettime() из библиотеки <time.h>. Так же по результатам проведенных исследований была подтверждена разница по времени между разными уровнями компиляции. Для значения N1 лучшее время показал уровень оптимизации -O3. Результаты других уровней отличаются на миллисекунды. Для больших значений N2 и N3 уровень оптимизации -Ofast показал лучший результат, существенно отличающийся от других уровней.

## Приложение 1. Исходный код программы

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <time.h>

double PiByGregoryLeibniz(long long int N){

    double pi = 1.0;
    double pow = -1;
    for (long long int i = 1; i <= N; ++i){
<------>pi += pow / (2 * i + 1);
<------>pow *= (-1);
    }    pi *= 4;
    return pi;
}

void TestClockGettime(long long int N){
    struct timespec start, end;
    clock_gettime(CLOCK_MONOTONIC_RAW, &start);
    double pi = PiByGregoryLeibniz(N);
    clock_gettime(CLOCK_MONOTONIC_RAW, &end);
    printf("Pi: %f\n", pi);
    printf("Time taken by clock_gettime: %lf sec.\n",\
end.tv_sec - start.tv_sec + 0.000000001*(end.tv_nsec - start.tv_nsec));
....
}

int main(int argc, char** argv){
    long long int N = 0;
    N = atoll(argv[1]);
    double pi = PiByGregoryLeibniz(N);
    printf("Pi: %f\n", pi);

    return 0;
}
```

## Приложение 2. Проверка программы на тестовых значениях N

```
evmpu@comrade:~/21204/osipov/lab1$ ./lab1.out 0
Pi: 4.000000
evmpu@comrade:~/21204/osipov/lab1$ ./lab1.out 1
Pi: 2.666667
evmpu@comrade:~/21204/osipov/lab1$ ./lab1.out 100
Pi: 3.151493
evmpu@comrade:~/21204/osipov/lab1$ ./lab1.out 1000
Pi: 3.142592
evmpu@comrade:~/21204/osipov/lab1$ ./lab1.out 1000000
Pi: 3.141594
```

## Приложение 3. Подбор необходимого значения N0

```
evmpu@comrade:~/21204/osipov/lab1$ ./lab1.out 5000000000
Pi: 3.141593
Time taken by clock_gettime: 34.755013 sec.
```

## Приложение 4. Строки команд компиляции на разных уровнях

```
evmpu@comrade:~/21204/osipov/lab1$ g++ -O0 lab1.cpp -o lab1_0.out
evmpu@comrade:~/21204/osipov/lab1$ g++ -O1 lab1.cpp -o lab1_1.out
evmpu@comrade:~/21204/osipov/lab1$ g++ -O2 lab1.cpp -o lab1_2.out
evmpu@comrade:~/21204/osipov/lab1$ g++ -O3 lab1.cpp -o lab1_3.out
evmpu@comrade:~/21204/osipov/lab1$ g++ -Os lab1.cpp -o lab1_s.out
evmpu@comrade:~/21204/osipov/lab1$ g++ -Ofast lab1.cpp -o lab1_fast.out
evmpu@comrade:~/21204/osipov/lab1$ g++ -Og lab1.cpp -o lab1_g.out
evmpu@comrade:~/21204/osipov/lab1$ ls
```

## Приложение 5. Результаты измерений времени работы программы на разных уровнях компиляции

1) Базовый уровень

```
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1.out 2500000000
Pi: 3.141593

real    0m17,591s
user    0m17,587s
sys     0m0,000s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1.out 5000000000
Pi: 3.141593

real    0m34,762s
user    0m34,755s
sys     0m0,000s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1.out 7500000000
Pi: 3.141593

real    0m52,006s
user    0m51,993s
sys     0m0,004s
```

2) -O0

```
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_0.out 2500000000
Pi: 3.141593

real    0m17,504s
user    0m17,501s
sys     0m0,000s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_0.out 5000000000
Pi: 3.141593

real    0m34,753s
user    0m34,747s
sys     0m0,000s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_0.out 7500000000
Pi: 3.141593

real    0m52,003s
user    0m51,990s
sys     0m0,004s
```

3) -O1

```
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_1.out 2500000000
Pi: 3.141593

real    0m17,505s
user    0m17,498s
sys     0m0,004s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_1.out 5000000000
Pi: 3.141593

real    0m34,752s
user    0m34,741s
sys     0m0,004s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_1.out 7500000000
Pi: 3.141593

real    0m52,263s
user    0m52,252s
sys     0m0,000s
```

4) -O2

```
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_2.out 2500000000
Pi: 3.141593

real    0m17,508s
user    0m17,504s
sys     0m0,000s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_2.out 5000000000
Pi: 3.141593

real    0m34,754s
user    0m34,747s
sys     0m0,000s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_2.out 7500000000
Pi: 3.141593

real    0m52,009s
user    0m51,995s
sys     0m0,004s
```

5) -Ofast

```
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_fast.out 2500000000
Pi: 3.141593

real    0m17,506s
user    0m17,498s
sys     0m0,004s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_fast.out 5000000000
Pi: 3.141593

real    0m34,512s
user    0m34,505s
sys     0m0,000s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_fast.out 7500000000
Pi: 3.141593

real    0m51,764s
user    0m51,751s
sys     0m0,004s
```

6) -Og

```
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_g.out 2500000000
Pi: 3.141593

real    0m17,507s
user    0m17,503s
sys     0m0,000s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_g.out 5000000000
Pi: 3.141593

real    0m34,753s
user    0m34,742s
sys     0m0,004s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_g.out 7500000000
Pi: 3.141593

real    0m52,012s
user    0m51,999s
sys     0m0,004s
```

7) -Os

```
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_g.out 2500000000
Pi: 3.141593

real    0m17,507s
user    0m17,503s
sys     0m0,000s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_g.out 5000000000
Pi: 3.141593

real    0m34,753s
user    0m34,742s
sys     0m0,004s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_g.out 7500000000
Pi: 3.141593

real    0m52,012s
user    0m51,999s
sys     0m0,004s
```

8) -O3

```
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_3.out 2500000000
Pi: 3.141593

real    0m17,419s
user    0m17,415s
sys     0m0,000s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_3.out 5000000000
Pi: 3.141593

real    0m34,689s
user    0m34,682s
sys     0m0,000s
evmpu@comrade:~/21204/osipov/lab1$ time ./lab1_3.out 7500000000
Pi: 3.141593

real    0m54,053s
user    0m54,035s
sys     0m0,000s
```