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 ЛАБОРАТОРНОЙ РАБОТЫ

«ИЗУЧЕНИЕ ОПТИМИЗИРУЮЩЕГО КОМПИЛЯТОРА»

студента 2 курса, 21211 группы

Михалёв Макар Андреевич

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Кудинов Антон Юрьевич

Новосибирск 2022

Цели работы

1. Изучение основных функций оптимизирующего компилятора, и некоторых примеров оптимизирующих преобразований и уровней оптимизации.

2. Получение базовых навыков работы с компилятором GCC.

3. Исследование влияния оптимизационных настроек компилятора GCC
на время исполнения программы.

Исследования

В ходе работы при компиляции программы использовались все возможные уровни оптимизации компилятора и было достигнуто время выполнения алгоритма: 30-60 секунд.

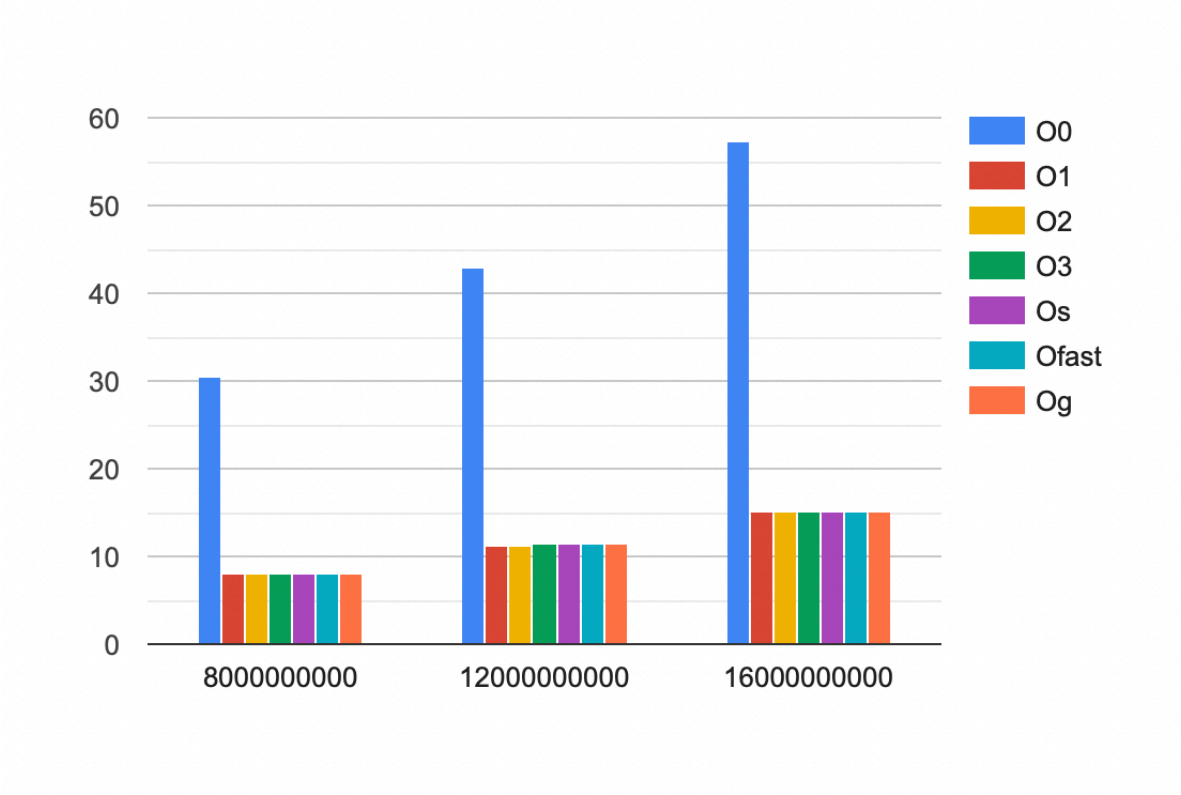
Вариант задания

Написать программу на языке C++, которая реализует

вычисления алгоритма числа Пи с помощью разложения в ряд (ряд Грегори-Лейбница) по формуле Лейбница N первых членов ряда

Графики зависимости времени выполнения программы

Уровень оптимизации\N	8000000000	12000000000	16000000000
O0	30.574498 sec	43.041584 sec	57.274578 sec
O1	8.057912 sec	11.317927 sec	15.174665 sec
O2	8.050860 sec	11.328789 sec	15.205110 sec
O3	8.059396 sec	11.404651 sec	15.167388 sec
Os	8.096891 sec	11.414411 sec	15.089054 sec
Ofast	8.067881 sec	11.402126 sec	15.069054 sec
Og	8.081210 sec	11.401525 se	15.170811 sec



Код:

```
#include <iostream>
#include <ctime>

double  SerExpanGregoryLeibniz()
{
    long long N = 16000000000;
    double res = 0, signDef = 1.0;
    for(long long i = 0; i < N; ++i)
    {
        res += (4.0 * signDef / (2.0 * (i) + 1.0));
        signDef *= (-1);
    }
    return res;
}

int main(int argc, const char * argv[]) {

    struct timespec start, end;

    clock_gettime(CLOCK_MONOTONIC_RAW, &start);


    double res = SerExpanGregoryLeibniz();
```

```
    clock_gettime(CLOCK_MONOTONIC_RAW, &end);

    printf("Time taken: %lf sec.\n", end.tv_sec-start.tv_sec +
0.000000001*(end.tv_nsec - start.tv_nsec));
    printf("%lf", res);
    return 0;
}
```

Команды, которые использовались для данной программы:

```
g++ -O0 Clock_GetTime.cpp -o c.bin -Wall
g++ -O1 Clock_GetTime.cpp -o c.bin -Wall
g++ -O2 Clock_GetTime.cpp -o c.bin -Wall
g++ -O3 Clock_GetTime.cpp -o c.bin -Wall
g++ -Os Clock_GetTime.cpp -o c.bin -Wall
g++ -Ofast Clock_GetTime.cpp -o c.bin -Wall
g++ -Og Clock_GetTime.cpp -o c.bin -Wall
```

Вывод:

Результат исследования был подтвержден так как уровень оптимизации способны уменьшать время работы, как видно из исследования уровень оптимизации -O0 не изменяет время работы программы, остальные уровни оптимизации показывают примерно одно время работы программы.Программа работает корректно поэтому вероятнее всего остальные уровни не задействовали своих возможностей. Также исследования показывают, что использование уровни оптимизаций ускоряет расколоть программы в разы.