# ЦЕЛЬ

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.

# ЗАДАНИЕ

1. Написать программу на языке C или C++, которая реализует выбранный алгоритм из задания.

2. Проверить правильность работы программы на нескольких тестовых наборах входных данных.

3. Выбрать значение параметра N таким, чтобы время работы программы было порядка 30-60 секунд.

4. Программу скомпилировать компилятором GCC с уровнями оптимизации -O0, -O1, -O2, -O3, -Os, -Ofast, -Og под архитектуру процессора x86.

5. Для каждого из семи вариантов компиляции измерить время работы программы при нескольких значениях N

# ОПИСАНИЕ РАБОТЫ

**Вариант задания**

Для выполнения мной было выбрано задание №6: Алгоритм вычисления функции ln(1+x) с помощью разложения в ряд по первым N членам этого ряда.

Реализация алгоритма:

#include <stdio.h>

double Power(double number, double pow) {

    double result = 1;

    for (int i = 1; i <= pow; i++) {

        result \*= number;

    }

    return result;

}

int main() {

    double n = 50000;

    double x = 0.5;

    double sum = 0;

    for (double i = 1; i <= n; i++) {

        sum += (Power(-1, i + 1) \* Power(x, i) / i);

    }

    printf("%f\n", sum);

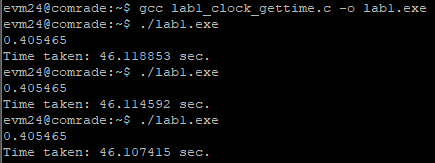
    return 0;

}

Правильность работы программы на нескольких тестовых наборах входных данных была проверена в отчете к предыдущей лабораторной работе.

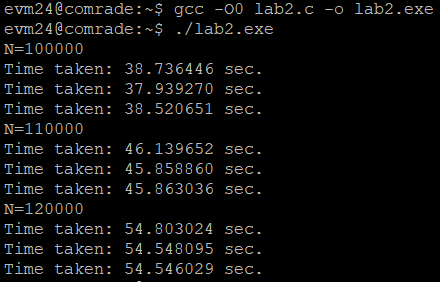
**Графики зависимости времени выполнения программы с уровнями оптимизации -O0, -O1, -O2, -O3, -Os, -Ofast, -Og от параметра N.**

Согласно заданию, время работы программы должно занимать от 30 до 60 секунд. Без указания компилятору применить определенный уровень компиляции программа работает 46.107415 секунд при N = 110000. Для каждого из вариантов компиляции будем проверять значения времени при N=110000±10000.

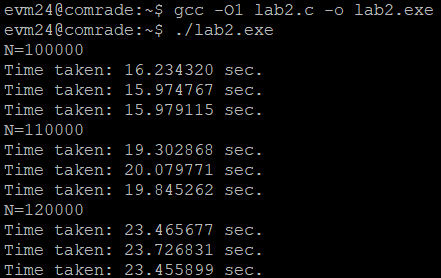


*Измерение времени для опрделения значения параметра N*

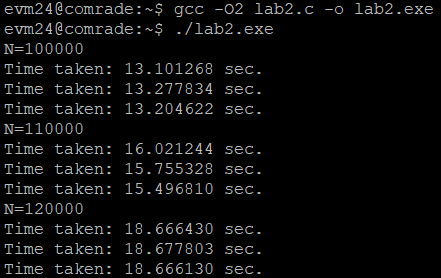
**-O0**

******

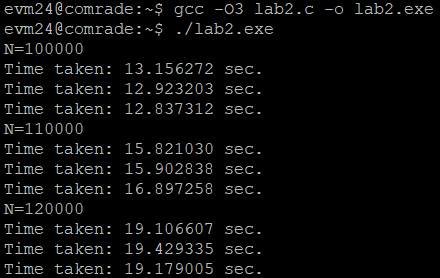
**-O1**

******

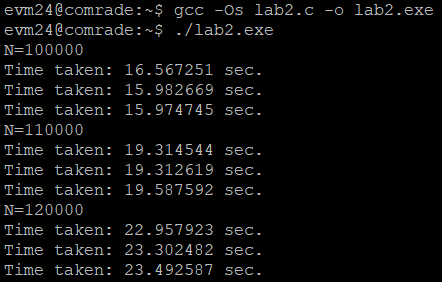
**-O2**



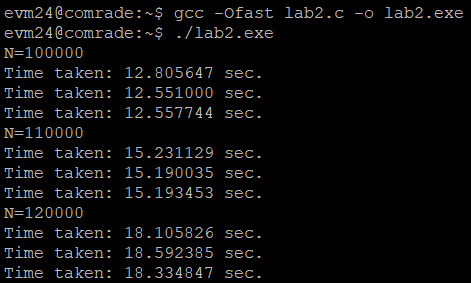
**-O3**

******

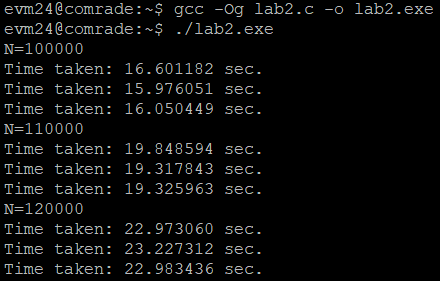
**-Os**

******

**-Ofast**



**-Og**



Сводная таблица результатов измерений (в секундах):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | O0 | O1 | O2 | O3 | Os | Ofast | Og |
| 100000 | 37,93927 | 15,974767 | 13,101268 | 12,837312 | 15,974745 | 12,551 | 15,976051 |
| 110000 | 45,85886 | 19,302868 | 15,49681 | 15,82103 | 19,312619 | 15,190035 | 19,317843 |
| 120000 | 54,546029 | 23,455899 | 18,66613 | 19,106607 | 22,957923 | 18,105826 | 22,97306 |

ЗАКЛЮЧЕНИЕ

Я изучила уровни оптимизации компилятора GCC, получила основные навыки работы с ним, а также исследовала влияние оптимизационных настроек на время исполнения программы.

При измерении времени без намеренного использования какого-либо из уровней оптимизации я получила значение, близкое к значению -O0.

По графику видим, что любой из уровней оптимизации заметно уменьшил время исполнения программы по сравнению с -O0.

По графику зависимости времени выполнения программы видно, что для выбранного мной алгоритма быстрее всего работают уровни оптимизации -O2, -O3 и -Ofast, самым быстрым из которых, обращаясь к таблице значений, является -Ofast.