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

«название работы»

студентки 2 курса, 24202 группы

Корсун Дарьи Андреевны

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
Кандидат технических наук
В.А.Перепёлкин

Новосибирск 2025

СОДЕРЖАНИЕ

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**.**
6. Составить отчет по лабораторной работе. Отчет должен содержать следующее.

ОПИСАНИЕ РАБОТЫ

Для изучения функций оптимизирующего компилятора была написана программа, реализующая подсчет значения синуса через разложение в ряд Тейлора. Она в свою очередь подходит для изучения работы компиляторы, так как потенциально содержит места, подлежащие изменению при разных уровнях оптимизации (циклы, которые можно раскрутить, функции которые могут напрямую подставлены в место их вызова, небольшой размер вычислений, позволяющий записать их на регистр для ускорения расчетов)

Код программы, реализующий вычисления функции sin x с помощью разложения в степенной ряд по первым N членам этого ряда:

$$\sin x = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots + \frac{(-1)^{n-1}}{(2n-1)!} x^{2n-1} + \cdots$$

Область сходимости ряда: $-\infty \leq x \leq \infty$
Код программы:

```c
#include <math.h>
#include <stdio.h>
#include <time.h>

double CalculateSin(double x, long N) {
  double term = x;
  double result = x;
  for (long n = 2; n <= N; ++n) {
    term *= -(x * x) / ((2 * n - 1) * (2 * n - 2));
    result += term;
  }
  return result;
}

int main() {
  struct timespec start, end;
  double x = M_PI / 3;
  // long N = 30000000000;
             // 63,4 / 22,6 / 22,4 / 22,4 / 22,4 / 95,4 / 22,5
  // long N = 25000000000;
             // 52,3 / 18,7 / 18,7 / 18,6 / 18,7 / 79,5 / 18,7
  // long N = 20000000000;
             // 41,3 / 15,0 / 15,0 / 14.9 / 14,9 / 63,3 / 15,0
  long N = 15000000000;
             // 31,7 / 11,2 / 11,2 / 11,2 / 11,2 / 47,7 / 11,2

  double result;
  clock_gettime(CLOCK_MONOTONIC_RAW, &start);

  result = CalculateSin(x, N);

  clock_gettime(CLOCK_MONOTONIC_RAW, &end);

  double time_taken =
      (end.tv_sec - start.tv_sec) +
                    0.000000001 * (end.tv_nsec - start.tv_nsec);

  printf("sin(%f) = %.12f\n", x, result);
  printf("time_taken = %.6f \n", time_taken);

  return 0;
}
```

Программы консоли для компиляции кода:

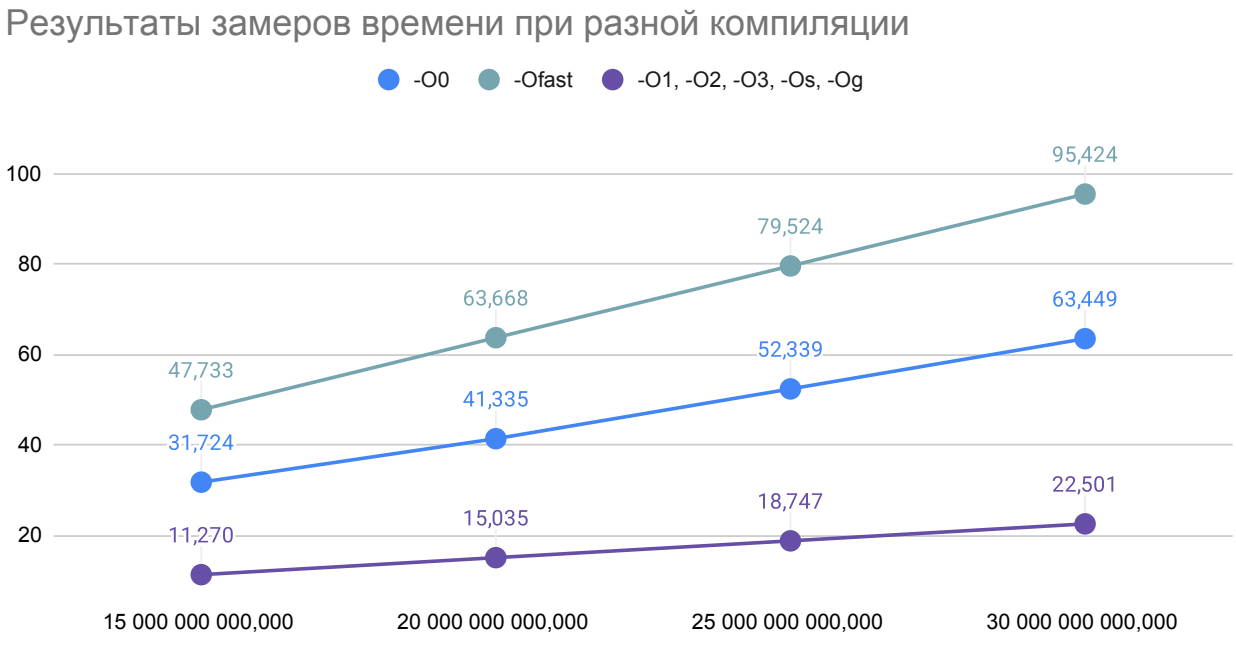
```
gcc -O0 lab2_sin.c -o lab2_sin
gcc -O1 lab2_sin.c -o lab2_sin
gcc -O2 lab2_sin.c -o lab2_sin
gcc -O3 lab2_sin.c -o lab2_sin
gcc -Os lab2_sin.c -o lab2_sin
gcc -Ofast lab2_sin.c -o lab2_sin
gcc -Og lab2_sin.c -o lab2_sin
```

В качестве N были подобраны значения (15000000000, 20000000000, 25000000000, 30000000000), которые при -O0 выполняются соответственно за (31.7, 41.3, 52.3, 63.4).

Приведем данные измерения в таблицу

Уровень оптимизации / N	15000000000	20000000000	25000000000	30000000000
-O0	31.724	41.335	52.339	63.449
-O1	11.269	15.010	18.756	22.637
-O2	11.287	15.007	18.731	22.473
-O3	11.269	14.953	18.686	22.468
-Os	11.278	14.956	18.737	22.456
-Ofast	47.733	63.668	79.524	95.424
-Og	11.270	15.035	18.747	22.501

Графики зависимости времени выполнения программы с уровнями оптимизации:



ЗАКЛЮЧЕНИЕ

Изучены основных функций оптимизирующего компилятора, и некоторых примеров оптимизирующих преобразований и уровней оптимизации. Получение базовых навыков работы с компилятором GCC. Проведено исследование влияния оптимизационных настроек компилятора GCC на время исполнения программы.

После проведенного исследования, можно прийти к выводу, что уровни оптимизации заложенные в компиляторы во многом ускоряют работу программы, при этом программа разобранная в данной лабораторной не дала сильных изменений на (-O1, -O2, -O3, -Os, -Og) уровнях оптимизации, что может свидетельствовать о том, что эти уровни компиляции настроены на обработку более сложных случаев, которые в данном коде, из-за его простоты, не встречаются.