###### 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, 23201 группы

**Смирнова Гордея Андреевича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

А.С.Матвеев

Новосибирск 2024

**СОДЕРЖАНИЕ**

[ЦЕЛИ 3](#_gjdgxs)

[ЗАДАНИЕ 3](#_30j0zll)

[ОПИСАНИЕ РАБОТЫ 4](#_1fob9te)

[ЗАКЛЮЧЕНИЕ](#_3znysh7) 6

Приложение 1 7

Приложение 2 8

# ЦЕЛИ

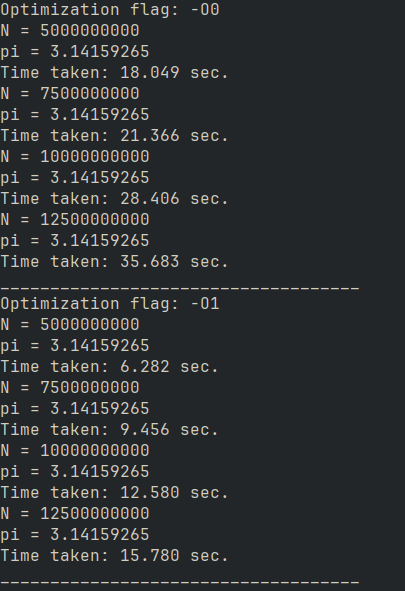
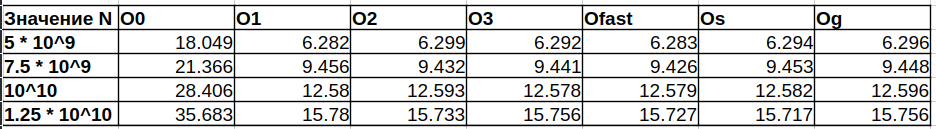
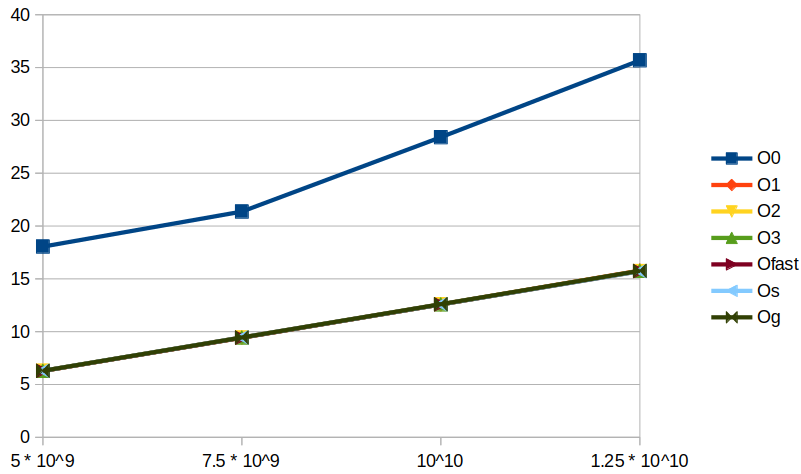
1. Изучение основных функций оптимизирующего компилятора, и некоторых примеров оптимизирующих преобразований и уровней оптимизации.
2. Получение базовых навыков работы с компилятором GCC.
3. Исследование влияния оптимизационных настроек компилятора GCC на время исполнения программы.

# 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.

# ОПИСАНИЕ РАБОТЫ

* 1. 1. На первом этапе был реализован алгоритм подсчёта числа Пи с помощью разложения в ряд (ряд Грегори-Лейбница) по формуле Лейбница N первых членов ряда: на языке программирования С и проверена корректность получаемых значений.

1. 2. Следующим шагом был добавлен замер времени для четырёх различных значений N.
2. 3. Произведен замер времени работы программы при разных ключах компиляции
3. 
4. 4. Данные о времени занесены в таблицу
5. 
6. 5. Построены графики зависимости времени от оптимизации компиляции и параметра N. (Рисунок 1)
7. 

**ЗАКЛЮЧЕНИЕ**

В ходе данной лабораторной работы мы увидели, что использование более высоких уровней оптимизации не всегда целесообразно, так как оптимального значения времени выполнения программы на данном примере мы достигаем уже при компилировании с ключом -О1.

**Приложение 1: *Исходный код программы (main.c) для тестирования***

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #include <unistd.h>  double PiLeibniz(long long const n) {  double result = 0;  for (long long i = 0; i < n; i++) {  double t = 1.0 / (2.0 \* i + 1.0);  if (i % 2 != 0) t = -t;  result += t;  }  return result \* 4;  }  int main(int argc, char \*argv[]) {  if (argc != 2) {  printf("Wrong args count\n");  return 1;  }  long long const n = atoll(argv[1]);  double result = 0;  struct timespec start, end;  clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);  result = PiLeibniz(n);  clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);  double const time = end.tv\_sec - start.tv\_sec + 0.000000001 \* (end.tv\_nsec - start.tv\_nsec);  printf("pi = %.8lf\n", result);  printf("Time taken: %.3lf sec.\n", time);  return 0;  } |

**Приложение 2: *Исходный код bash-скрипта для тестирования***

|  |
| --- |
| #!/bin/bash  n\_arr=(5000000000 7500000000 10000000000 12500000000)  for oflag in "-O0" "-O1" "-O2" "-O3" "-Ofast" "-Os" "-Og"  do  echo "Optimization flag: $oflag"  gcc main.c $oflag -o main.out  for i in "${n\_arr[@]}"  do  echo "N = $i"  ./main.out $i  done  echo "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"  done |