###### 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, 22201 группы

**Рабецкий Валерий Дмитриевич**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

А.С.Матвеев

Новосибирск 2023

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_gjdgxs)

[ЗАДАНИЕ 3](#_30j0zll)

[ОПИСАНИЕ РАБОТЫ 4](#_1fob9te)

[ЗАКЛЮЧЕНИЕ 7](#_3znysh7)

Приложение 1 4

Рисунок 1 5

Рисунок 2 6

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.

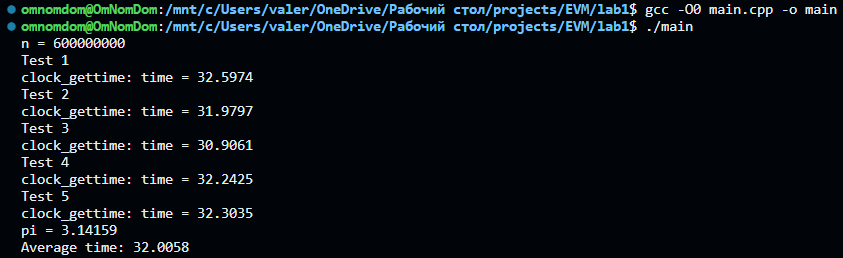
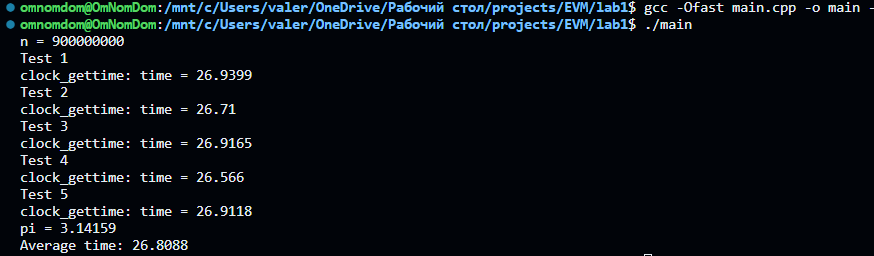
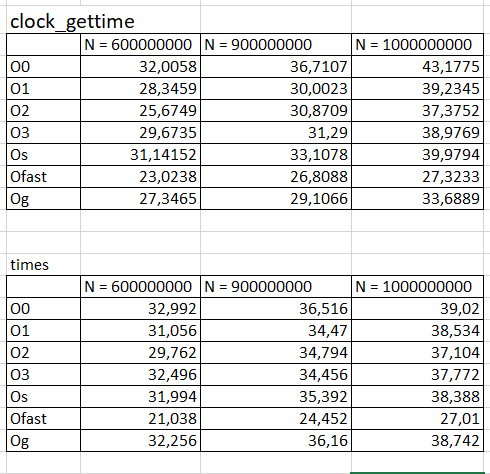
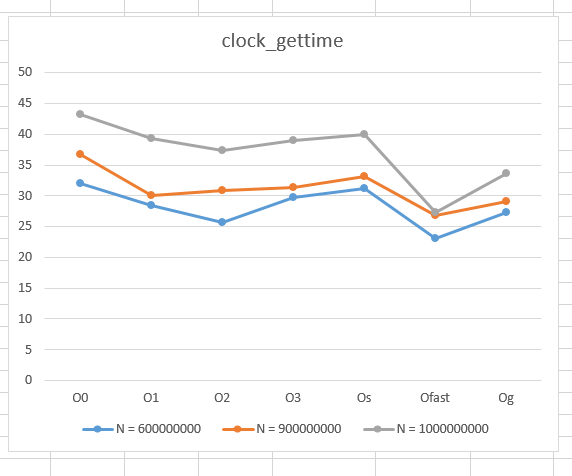
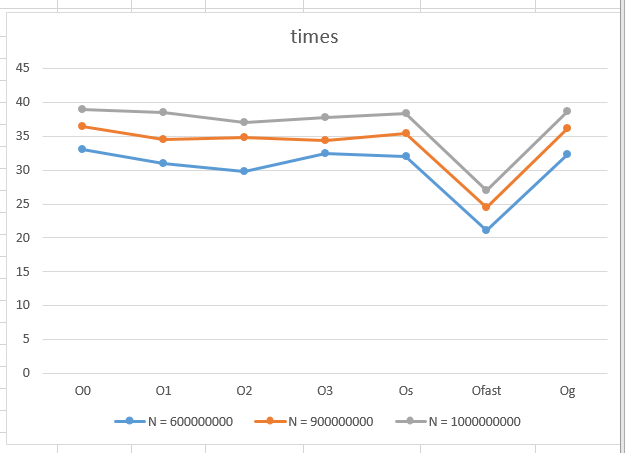
# ЗАДАНИЕ

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 трёх различных значений N. (Приложение 1)
2. 3. Произведен замер времени работы программы при разных ключах компиляции
3. 
4. 
5. 4. Данные о времени занесены в таблицу
6. 5. Построены графики зависимости времени от оптимизации компиляции и параметра N. (Рисунок 1 и 2)
7. 
8. 

**ЗАКЛЮЧЕНИЕ**

В ходе данной лабораторной работы мы увидели, что использование небезопасных режимов компиляции не всегда оправданно и дает наилучший результат, оптимального значения времени мы достигаем уже при компилировании программы с ключом -О2.

**Приложение 2: *Исходный код программы для тестирования***

|  |
| --- |
| #include <iostream>  #include <cmath>  #include <ctime>  #include <sys/times.h>  #include <unistd.h>  using std::cout;  using std::endl;  int main()  {  double averageTime = 0;  double pi;  long long n = 900000000;  cout << "n = " << n << endl;  for (int j = 0; j < 5; ++j)  {  pi = 0;  double temp;  // struct tms start, end;  // long long clocks\_per\_sec = sysconf(\_SC\_CLK\_TCK);  // // cout << "Частота тактов ядра в сек = " << clocks\_per\_sec << endl;  // long long clocks;  // times(&start);  timespec start\_gettime, end\_gettime;  clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start\_gettime);  for (int i = 0; i < n; i++)  {  temp = ((pow(-1, i)) / (2 \* i + 1));  pi += temp;  }  pi = 4 \* pi;  // times(&end);  // clocks = end.tms\_utime - start.tms\_utime;  cout << "Test " << j + 1 << endl;  // cout << "Time taken: " << (double)clocks / clocks\_per\_sec << endl;  // averageTime += (double)clocks / clocks\_per\_sec;  clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end\_gettime);  cout << "clock\_gettime: time = " << end\_gettime.tv\_sec - start\_gettime.tv\_sec + 0.000000001\*(end\_gettime.tv\_nsec - start\_gettime.tv\_nsec) << endl;  averageTime += end\_gettime.tv\_sec - start\_gettime.tv\_sec + 0.000000001\*(end\_gettime.tv\_nsec - start\_gettime.tv\_nsec);  }  cout << "pi = " << pi << endl;  cout << "Average time: " << averageTime / 5 << endl;  return 0;  } |