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 методики измерения времени работы подпрограммы.
2. Изучение приемов повышения точности измерения времени работы подпрограммы.
3. Изучение способов измерения времени работы подпрограммы.
4. Измерение времени работы подпрограммы в прикладной программе.

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 15 секунд.
4. По приведенной методике определить время работы подпрограммы тестовой программы с относительной погрешностью не более 1%.
5. Составить отчет по лабораторной работе.

ОПИСАНИЕ РАБОТЫ.

В ходе выполнения лабораторной работы, для изучения методов измерение работы времени подпрограммы был выбран четвертый вариант задания:

Алгоритм 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: $-\infty \leq x \leq \infty$

Таймером для измерения времени работы подпрограммы был выбран: 4.2 Библиотечная функция clock_gettime. Функция clock gettime с параметром CLOCK_MONOTONIC_RAW сохраняет значение системного таймера в структуру struct timespec. Структура состоит из двух полей: tv_sec и tv_nsec (можно считать их тип long int), задающих количество секунд и наносекунд (10−9 сек.), прошедших с некоторого неспецифицированного момента времени в прошлом. Обоснование оптимальности выбора:

1. «Утилитой time», выбранный выше метод позволяет исключить из времени работы подпрограммы стадии инициализации и загрузки библиотек, что является достаточным преимуществом.
2. «Библиотечная функция `times`», предлагает более низкую точность по сравнению с выбранным методом из-за округления до кванта времени переключения процессов (4 мс), в то время как выбранная функция даёт погрешность на 45 порядком меньше.
3. «Машинная команда rdtsc», потенциально может выдавать более точные значения чем выбранная функция, но из-за динамической изменения тактовой частоты на ноутбуках с чипами Apple M4, импользование этой функции становится не эффективной из-за большой погрешности.

В итоге выбранный таймер является оптимальным за счет маленькой погрешности и возможности измерять только выполнения конкретной подпрограммы, исключая инициализацию и загрузки библиотек. Недостаток связанный с тем, что измеренный интервал времени включает в себя работу других процессов нивелируется маленькой нагрузкой на ноутбук и возможностью свести сторонние процессы к минимуму.

После выбора таймер для измерения работы подпрограммы был написан код на языке программирования C++ (прил. 1), реализующий выбранный алгоритм. В качестве компилятора используется gcc версии 17.0.0.

Правильность работы программы была проверена путем сравнения результатов вычисления с эталонными для данного угла. При N>10 программа выдаёт правильный ответ с точностью больше девяти знаком после запятой. Ниже прилагаю таблицу с полученными результами для разных значений N.

N, кол-во членов разложения	Полученное значение синуса
1	1.047197551197
5	0.866025445100
10	0.866025403784
1000	0.866025403784

Эталонное значение, взятое со сторонних ресурсов: 0.86602540378444

Для увеличения времени работы программы до более чем 15 секунд было подобранно N = 8000000000.

Для увеличение точности были выполнены действия предложенные в задании, результат их влияния на время работы программы можно видеть в таблице ниже.

Действие для уточнения времени работы.	Результат измерения, с.
Отсутствие действия для улучшения	17.445255
Сброс буфера отложенной записи на диске	16.994185
Уменьшения влияния сторонних процессов	16.626042

Так как время самой программы всегда остаётся неизменным. Наименьшее время значение и будет наиболее точным, так как посторонние факторы только увеличивают все работы программа, следовательно минимизация этих факторов и даёт наилучший результат.

Посчитаем относительную погрешность полученных измерений по формуле: $\frac{6.0 \times 10^{-8}}{1.6626042} \times 100\,\% = 3.608 \times 10^{-7}\,\%$ что меньше 1% как и требуется в задании.

ЗАКЛЮЧЕНИЕ

Изучены методики измерения времени работы подпрограммы, приемы повышения точности измерения времени работы подпрограммы, способы измерения времени работы подпрограммы. Также было проведено измерение времени работы подпрограммы в прикладной программе на примере вычисления синуса с помощью разложения в ряд Тейлора.

Экспериментально показана эффективность приемов по повышению точности измерения времени (сброс буфера отложенной записи на бирке и уменьшение влияния сторонних процессов).

Приложение 1. *Код программы на C*

Команда для компиляции:
gcc lab1_sin.c -o lab1_sin

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 = 8000000000;

  double result;
  clock_gettime(CLOCK_MONOTONIC, &start);

  result = CalculateSin(x, N);

  clock_gettime(CLOCK_MONOTONIC, &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;
}
```