###### 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

«ОПРЕДЕЛЕНИЕ ВРЕМЕНИ РАБОТЫ ПРИКЛАДНЫХ ПРОГРАММ»

студента 2 курса, 22202 группы

**Бальчинова А.С.**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

В.А. Перепёлкин

Новосибирск 2023

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc18443921)

[ЗАДАНИЕ 3](#_Toc18443922)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc18443923)

[ЗАКЛЮЧЕНИЕ 5](#_Toc18443924)

[Приложение. *Листинг программы* 6](#_Toc18443925)

# ЦЕЛИ

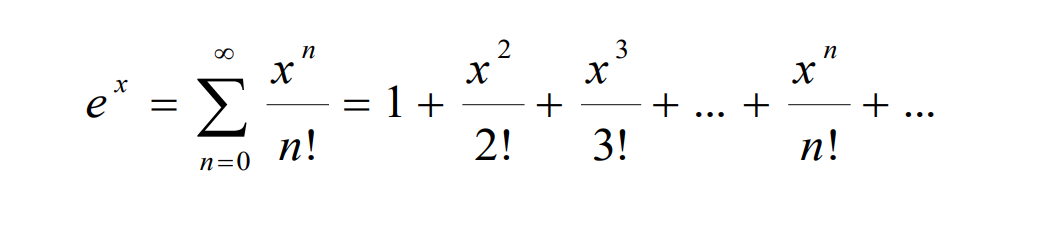
1. Изучение методики измерения времени работы подпрограммы.

2. Изучение приемов повышения точности измерения времени работы подпрограммы.

3. Изучение способов измерения времени работы подпрограммы.

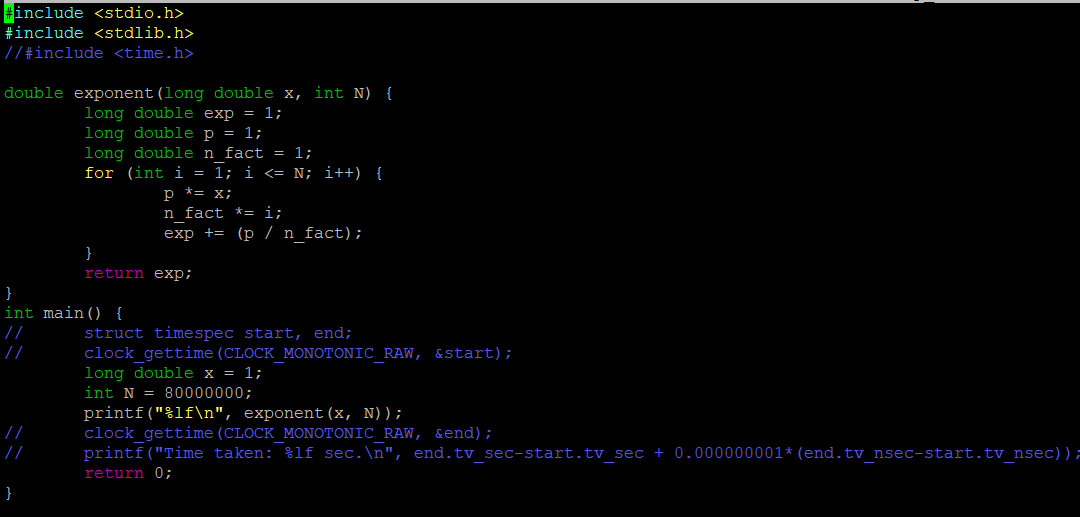
4. Измерение времени работы подпрограммы в прикладной программе.

# ЗАДАНИЕ

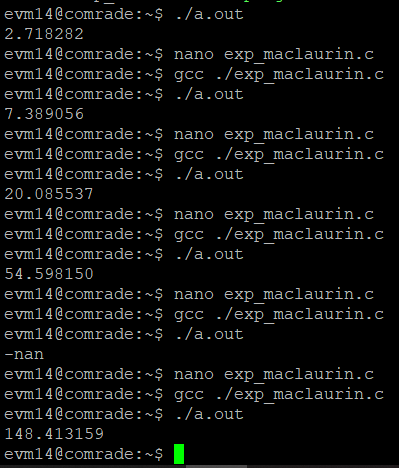
Алгоритм вычисления функции ex с помощью разложения в ряд Маклорена по первым N членам этого ряда: 

Область сходимости ряда: − ∞ ≤ x ≤ ∞ .

# ОПИСАНИЕ РАБОТЫ

1. Реализация алгоритма из задания на языке программирования C. 
2. Проверка правильности работы программы на нескольких тестовых наборах входных данных.
3. x = 1, N = 80000, ответ e^x = e^1 = 2.718282
4. x = 2, N = 80000, ответ e^x = e^2 = 7.389056
5. x = 3, N = 80000, ответ e^x = e^3 = 20.085537
6. x = 4, N = 80000, ответ e^x = e^4 = 54.598150

Программа выдаёт правильный ответ на нескольких тестовых значениях x, от N зависит точность вычислений.



1. Выбор значение параметра N таким, чтобы время работы программы было порядка 15 секунд.

Экспериментально (подбором) было выбрано значение N = 80,000,000, при котором время работы программы составляет примерно 14.7 секунд.

1. По приведенной методике определить время работы подпрограммы тестовой программы с относительной погрешностью не более 1%.

Время работы программы определялось при помощи двух методик:

1. Основная методика - библиотечная функция “clock\_gettime”:

её точность зависит от точности системного таймера.

В ОС GNU Linux/UNIX: 1 нс (1∙10^(−9) с).

1. Дополнительная методика - утилита “time”: её точность определяется точностью системного таймера и точностью измерения времени работы процесса.

**Описание основной методики для определения времени работы программы:**

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) cек.), прошедших с некоторого неспецифицированного момента времени в прошлом. В приведённом примере сохраняется значение таймера перед выполнением кода и после него. Разница показаний преобразуется в секунды и выводится на экран. Кроме системного таймера, функция позволяет получать значения и других таймеров, например, времени процесса или потока. Подробнее об этом можно прочитать в документации к этой функции. Реализация функции clock\_gettime находится в библиотеке rt, поэтому при компиляции программы необходимо добавить ключ компиляции ‘-lrt’. Пример команды компиляции (в некоторых системах требуется ключ ‘-lrt’ писать в конце команды): user@host:~$ gcc prog.c –o prog –lrt .

Точность: зависит от точности системного таймера. Обычно в ОС Windows: 55 мс (55∙10−3 с), в ОС GNU Linux/UNIX: 1 нс (1∙10^(−9) с).

Достоинство: переносимость – вне зависимости от аппаратного обеспечения функция доступна пользователю, т.к. реализуется ОС.

Недостатки: относительно низкая точность (обычно ниже, чем у счётчика тактов, но выше, чем у функции times), и измеренный интервал включает время работы других процессов, которые работали на процессоре в измеряемый период.

**Результаты измерения времени работы программы:**

Утилита “time”.

Среднее время: 14,786

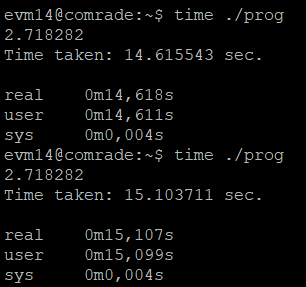
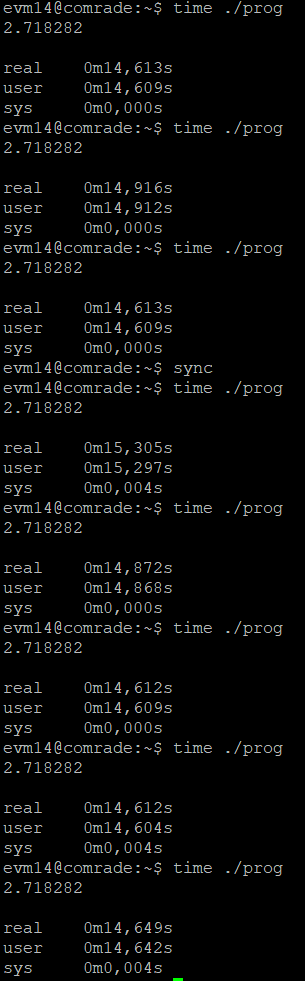
Минимальное время: 14,604

# Команда библиотеки time.h “clock\_gettime”.

Среднее время: 14,7232226

Минимальное время: 14,611691

Относительная погрешность: 0,000000001 с/ 14,604 с=0,00000000684%



# ЗАКЛЮЧЕНИЕ

В процессе работы была изучена методика измерения времени работы программы, а также приемы повышения точности измерения времени её работы. На практике была изучена функция “clock\_gettime”.

Используя методику многоразового измерения времени работы программы мы смогли вычислить время её работы с относительной погрешностью не превышающей 1% (точнее 0,00000000684%).

# Приложение. *Листинг программы*

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

double exponent(double x, int N) {

double exp = 1;

double p = 1;

double n\_fact = 1;

for (int i = 1; i <= N; i++) {

p \*= x;

n\_fact \*= i;

exp += (p / n\_fact);

}

return exp;

}

int main() {

struct timespec start, end;

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &start);

long double x = 1;

int N = 80000000;

printf("%lf\n", exponent(x, N));

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("Time taken: %lf sec.\n", end.tv\_sec-start.tv\_sec + 0.000000001\*(end.tv\_nsec-start.tv\_nsec));

return 0;

}