**ЗАДАНИЕ К ЛАБОРАТОРНОЙ РАБОТЕ №5**

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 5-10 секунд.

4. Определить время работы тестовой программы с относительной погрешностью не более 1% с помощью ***функции rdtsc и*** с помощью любой функции таймера системного времени или любой функции таймера времени процесса.

5. Зациклить программу и определять время работы кода в цикле 15 раз, печатать значение времени в файл. Выбрать минимальное значение времени и напечатать в файл. Напечатать в файл также результат расчетов своей программы.

6. Написать код на ассемблере, реализующий вариант задания. Сделать ассемблерную вставку в программу на С++. Выполнить пункты 2-5 для нового варианта программы.

7. Составить отчет по лабораторной работе. Отчет должен содержать

следующее:

**1.** Титульный лист.

**2.** Цель лабораторной работы.

**3.** Вариант задания.

**4.** Описание методики для определения времени работы

программы.

**5.** Результат измерения времени работы программы.

**6.** Полный компилируемый листинг реализованной программы и

команду для ее компиляции.

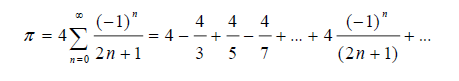
**7.** Вывод по результатам лабораторной работы. Указать:

* какие функции использовались для определения времени,
* в каких единицах измерения получили время,
* какая функция работает точнее,
* какой вариант программы выполняется быстрее: на ассемблере или на С++.

**ВАРИАНТЫ ЗАДАНИЙ**

**1.** Алгоритм вычисления числа Пи с помощью разложения в ряд (ряд

Грегори-Лейбница) по формуле Лейбница N первых членов ряда:



**2.** Алгоритм вычисления числа Пи метом Монте-Карло. Алгоритм состоит в

следующем. Сначала в квадрат с центром в начале координат и со

стороной два вписывается круг с единичным радиусом. Затем в этом

квадрате случайным образом с равномерным распределением

генерируются N точек. Точка может попасть в окружность или нет

(условие попадания x2 + y2 ≤ 1). Далее определяется число M точек,

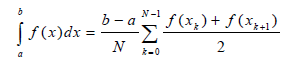
попавших в круг. При достаточно большом числе бросков N, по

значениям M и N вычисляется число Пи:

****

**3.** Алгоритм вычисления определенного интеграла сложной функции

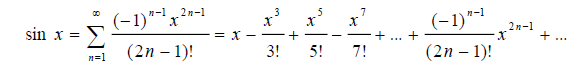
методом трапеций:



где f(x) = ex sin(x), a = 0, b = π, N - число интервалов.

**4.** Алгоритм вычисления функции sin x с помощью разложения в степенной

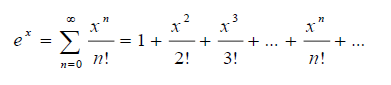
ряд по первым N членам этого ряда:



Область сходимости ряда: − ∞ ≤ *x* ≤ ∞ .

**5.** Алгоритм вычисления функции ex с помощью разложения в ряд

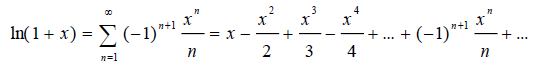
Маклорена по первым N членам этого ряда:



Область сходимости ряда: − ∞ ≤ *x* ≤ ∞ .

**6.** Алгоритм вычисления функции ln(1+x) с помощью разложения в ряд по

первым N членам этого ряда:



Область сходимости ряда: − 1 ≤ *x* ≤ 1 .

**7.** Алгоритм сортировки методом пузырька. Дан массив случайных чисел

длины N. На первой итерации попарно упорядочиваются все соседние

элементы; на второй – все элементы, кроме последнего элемента; на

третьей – все элементы, кроме последнего элемента и предпоследнего

элемента и т.п.

8. Алгоритм вычисления функции tg(x) с помощью разложения в ряд Маклорена по первым N членам этого ряда:



Область сходимости ряда: − π/2 ≤ *x* ≤ π/2 .

9. Алгоритм вычисления функции синус гиперболический sh(x) с помощью разложения в ряд по первым N членам этого ряда:



10. Алгоритм вычисления функции косинус гиперболический ch(x) с помощью разложения в ряд по первым N членам этого ряда:



.

**Теоретический материал.**

**Таймер системного времени**

GetSystemTime(), GetTickCount(), time() показывает реальное время в милисекундах.

**Таймер времени процесса**

GetThreadTimes(), GetProcessTimes(), Clock() показывают время выполнения процесса в милисек.

**Счетчик тактов процессора**

x86/x86-64: rdtsc  читает счётчик TSC (Time Stamp Counter) и возвращает в регистрах EDX:EAX 64-битное количество тактов с момента последнего сброса процессора.

Использование rdtsc в WINDOWS

#include <intrin.h>

#pragma intrinsic(\_\_rdtsc)

unsigned \_\_int64 rdtsc() {

return \_\_rdtsc();

}

unsigned \_\_int64 t1,t2,t;

t1=rdtsc();

t2=rdtsc();

t=(t2-t1)/FP; //FP-частота процессора

P.S.: Узнать количество процессоров в системе можно функцией GetSystemInfo():

SYSTEM\_INFO sysInfo;

GetSystemInfo( &sysInfo );

numCpus = sysInfo.dwNumberOfProcessors;

**Что должно быть в файле**

Имя исследуемой функции

Значения времени выполнения кода

Минимальное значение времени выполнения кода

Результат работы кода

**Пример файла**

GetTickCount

47

62

47

63

47

62

63

47

62

47

62

47

63

47

62

47

63

62

47

62

47

63

47

62

47

63

62

47

62

tmin=0.047

pi=3.14173

M=785433

N=100000

**КОНТРОЛЬНЫЕ ВОПРОСЫ**

∙ Назовите цели измерения времени.

∙ В чем состоит методика измерения времени работы программы?

∙ Назовите способы измерения времени работы программы. Перечислите их особенности.

∙ Каким способом лучше измерять большие промежутки времени (порядка нескольких часов)?

∙ Каким способом лучше измерять малые промежутки времени (порядка времени работы нескольких команд процессора)?\_\_