**Задание 3**

**«Вещественный тип. Приближенные вычисления.**

**Табулирование функций»**

**Оглавление**

[Постановка задачи 2](#__RefHeading___Toc3187_1941734421)

[Теоретическая часть 3](#__RefHeading___Toc3189_1941734421)

[Общий метод решения 4](#__RefHeading___Toc3191_1941734421)

[Функциональное назначение 5](#__RefHeading___Toc3193_1941734421)

[Описание программы 6](#__RefHeading___Toc3195_1941734421)

[Протокол 8](#__RefHeading___Toc3197_1941734421)

[Вывод 10](#__RefHeading___Toc3199_1941734421)

# Постановка задачи

Составить программу на Си, которая печатает таблицу значений

элементарной функции, вычисленной двумя способами: по формуле Тейлора

и с помощью встроенных функций языка программирования. В качестве

аргументов таблицы взять точки разбиения отрезка [a, b] на n равных частей

(n+1 точка включая концы отрезка), находящихся в рекомендованной

области хорошей точности формулы Тейлора. Вычисления по формуле

Тейлора проводить по экономной в сложностном смысле схеме с точностью ε

\* 10k, где ε - машинное эпсилон аппаратно реализованного вещественного

типа для данной ЭВМ, а k – экспериментально подбираемый коэффициент,

обеспечивающий приемлемую сходимость. Число итераций должно

ограничиваться сверху числом порядка 100. Программа должна сама

определять машинное ε и обеспечивать корректные размеры генерируемой

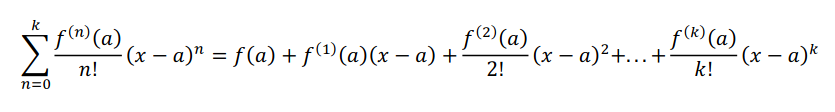
таблицы.

**Вариант №23**



# Теоретическая часть

**Формула Тейлора** — формула разложения функции в бесконечную сумму  
степенных функций. Формула широко используется в приближённых  
вычислениях, так как позволяет приводить трансцендентных функций к  
более простым. Сама она является следствием теоремы Лагранжа о среднем  
значении дифференцируемой функции. В случае a=0 формула называется  
рядом Маклорена.



**Машинное эпсилон** — числовое значение, меньше которого невозможно  
задавать относительную точность для любого алгоритма, возвращающего  
вещественные числа. Абсолютное значение для машинного эпсилон зависит  
от разрядности сетки применяемой ЭВМ и от разрядности используемых при  
расчёте чисел. Формально это машинное эпсилон определяют как число,  
удовлетворяющее равенству 1 + ε = 1. Фактически, два отличных от нуля  
числа являются равными с точки зрения машинной арифметики, если их  
модуль разности меньше или не превосходит машинное эпсилон.

# Общий метод решения

Общий метод решения заключается в нахождении значения функции в некоторой точке при помощи двух способов.

Первый способ заключается в использовании функций, встроенных в стандартную математическую библиотеку языка Си «math.c». В стандартной библиотеке имеется функция «atan(x)», которая считает арктангенс x, где x — единственный аргумент функции.

Основополагающей вещью в вычислении функции с помощью ряда Тейлора является наличие машинного эпсилон, которое является критерием точности вычислений на заданной ЭВМ.

Его можно найти путём сравнения 1 + ε с 1 (1 + ε == 1). Последнее число, при стремлении к нулю, при котором данное выражение выдаст false и будет машинным эпсилон.

Я буду вычислять на каждом шаге итерации n-ое слагаемое ряда Тейлора и, в случае если данное слагаемое будет меньше k\*ε(где k — эмперически-подобранный коэффицент), то далее вычислять ряд Тейлора является бессмысленным, так как члены ряда дошли до максимальной точности компьютера.

# Функциональное назначение

Программа предназначена для выполнения вещественных вычислений значений трансцедентных функций в алгебраической форме с использованием ряда Тейлора.

Ряд Тейлора – это разложение функции в бесконечную сумму степенных

функций. Если функция f(x) имеет непрерывные производные до (n + 1) порядка, то ее можно разложить по формуле Тейлора.

Ранее данный метод использовался для аппаратного вычисления подобных

функций, так как в то время компьютеры были способны только на сложение,

вычитание и умножение. На сегодняшний день аппаратное обеспечение позво-

ляет вычислять трансцендентные функции другими способами, которые более

эффективны во всех смыслах.

# Описание программы

Программа работы:

– Определяем стандартные функции языка С, подключая заголовки «math.h» и «stdio.h»

– Определяем функцию вычисления машинного эпсилон

– Определяем функцию для вычисления функции при помощи встроенных функций

– Определяем функцию для вычисления члена ряда Тейлора

– Вводим с клавиатуры количество разбиений отрезка [ a, b ] на части

– Вычисляем машинное эпсилон и выводим

– Печатаем таблицу аргументов функций, значений полученных средствами языка С и ряда Тейлора, количество итераций запрошенное машиной для вычисления значения функции

– Конец работы программы

Таблица 1. Описание переменных и констант

| long double k | Эмпирический коэффицент для eps |
| --- | --- |
| long double eps | Машинный эпсилон |
| long double a,b | Границы отрезка |
| int n | Кол-во итераций |
| int steps | Кол-во отрезков |
| int max\_iters | Максимальное кол-во итераций |
| long double cur\_member | I-ое слагаемое ряда |
| long double sum | Сумма ряда |

Таблица 2. Описание функций

|  |  |  |
| --- | --- | --- |
| Название функции | Входные аргументы | Описание функции |
| compute\_epsilon | - | Функция считает машинный epsilon, методом, описанным выше, а именно сравнивая 1+ε/2 и 1. Пока выражение 1 < 1 + ε/2 возвращает true, функция делит epsilon пополам. |
| inner\_func | long double x | Функция вычисляет функцию, данную в задаче при помощи встроенных в язык программирования С средств. Используется функция atan, которая вычисляет арктангенс входного числа x. |
| teilor\_member | long double x, int n | Функция вычисляет член ряда Тейлора. |

# Протокол

polina@pelis:~$ cat > kurs3.c

#include <stdio.h>

#include <math.h>

typedef long double ld;

const ld k = 10e2;

const ld a = 0.0;

const ld b = 0.5;

const int max\_iters = 100;

double compute\_epsilon()

{

double e = 1.0;

while(1 < 1 + e/2)

e /= 2;

return e;

}

ld inner\_func(ld x)

{

return atan(x);;

}

ld teilor\_member(ld x, int n)

{

ld v = 2\*(-1)\*(n&1)+1;

v \*= powl(x,2\*n+1);

v /= 2\*n+1;

return v;

}

int main()

{

ld steps;

scanf("%Lf",&steps);

ld step = (b-a)/steps;

double eps = compute\_epsilon();

printf("Machine epsilon for double for this system is %.20f\n\n", eps);

printf("\t<<<< Function f(x) = arctg(x) >>>>\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("| x | Sum | atan(x) | |\n");

printf("|\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|\n");

for(ld x = a; x <= b + step/2; x += step)

{

int n = 0;

ld cur\_member = 1;

ld sum = 0;

while((fabsl(cur\_member) > eps \* k && n < max\_iters) || n == 2)

{

cur\_member = teilor\_member(x, n);

sum += cur\_member;

n++;

}

printf("|%.2Lf|%.19Lf|%.19Lf|%3d|\n", x, sum, inner\_func(x), n);

}

printf("|\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|\n");

}

^C

polina@pelis:~$ gcc kurs3.c -lm

polina@pelis:~$ ./a.out

5

Machine epsilon for double for this system is 0.00000000000000022204

<<<< Function f(x) = arctg(x) >>>>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| x | Sum | atan(x) | |

|\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|

|0.00|0.0000000000000000000|0.0000000000000000000| 1|

|0.10|0.0996686524911620935|0.0996686524911620381| 7|

|0.20|0.1973955598498834214|0.1973955598498807751| 9|

|0.30|0.2914567944778639635|0.2914567944778670983| 12|

|0.40|0.3805063771123778197|0.3805063771123649019| 15|

|0.50|0.4636476090008437998|0.4636476090008060935| 19|

|\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|

polina@pelis:~$ ./a.out

12

Machine epsilon for double for this system is 0.00000000000000022204

<<<< Function f(x) = arctg(x) >>>>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| x | Sum | atan(x) | |

|\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|

|0.00|0.0000000000000000000|0.0000000000000000000| 1|

|0.04|0.0416425790985884835|0.0416425790985884212| 5|

|0.08|0.0831412318884405153|0.0831412318884412194| 6|

|0.12|0.1243549945467633041|0.1243549945467614382| 7|

|0.17|0.1651486774146234473|0.1651486774146268266| 8|

|0.21|0.2053953891897731590|0.2053953891897674078| 9|

|0.25|0.2449786631268539111|0.2449786631268641435| 10|

|0.29|0.2837941092083262921|0.2837941092083278716| 12|

|0.33|0.3217505543966465952|0.3217505543966421855| 13|

|0.37|0.3587706702705587040|0.3587706702705722450| 14|

|0.42|0.3947911196997541257|0.3947911196997615502| 16|

|0.46|0.4297622790967215155|0.4297622790966885153| 17|

|0.50|0.4636476090008437997|0.4636476090008060935| 19|

|\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_|

polina@pelis:~$

# Вывод

В процессе выполнения данного задания курсового проекта, я получила навыки вычисления значений функции с помощью ряда Тейлора и машинного эпсилон и его дальнейшего использования .

После генерации таблицы значений заданной функции можно увидеть, что значения совпадают до 14-17 знака после запятой. Из-за того, что существует понятие ограниченности разрядной сетки, вещественные числа имеют диапазон представления в памяти компьютера, что неизбежно приводит к тому, что в вычислениях в окрестности границ этого диапазона возникают погрешности.

На данный момент использование ряда Тейлора для вычисления трансцендентных функций является не оправданным, так как они требуют намного больше ресурсов, чем современные методы и имеют меньшую точность.