Московский Авиационный институт

(Национальный исследовательский университет)

**Курсовой проект**

**по курсам**

**“Архитектура компьютера”, “Программные и аппаратные средства информатики”**

**1 семестр**

**Задание 3**

Студент: Белоносов К.А.

Группа: М8О-103Б-21

Руководитель: Севастьянов В. С.

Оценка:

Дата:

Подпись:

**Содержание**

1. Введение 3
2. Вариант 3
3. Программное обеспечение 3
4. Описание работы, алгоритм 4
5. Описание переменных 4
6. Проверка программы 5
7. Вывод 6
8. Исходный код 6

**Введение**

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\*k, 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**

Вариант №3



**Программное обеспечение**

Операционная система семейства: linux, наименование: ubuntu, версия 20.04.3 LTS

Интерпретатор команд: bash версия 5.0.17(1)

Система программирования VS Code, редактор текстов emacs версия 27.1

**Описание работы, алгоритм**

Программа в цикле вычисляет значение функции с помощью ряда Тейлора в точках заданного отрезка. Вычисление происходит до тех пор, пока число итераций меньше 100 или значение очередного слагаемого станет меньше машинного эпсилона, умноженного на константу. Также вычисляется значение функции “обычным” методом.

По ходу вычислений печатается таблица. В первом столбце – текущий x, во второй – значение функции по ряду Тейлора, в третьей – значение самой функции, а в четвертой – число пройденных итераций ряда Тейлора.

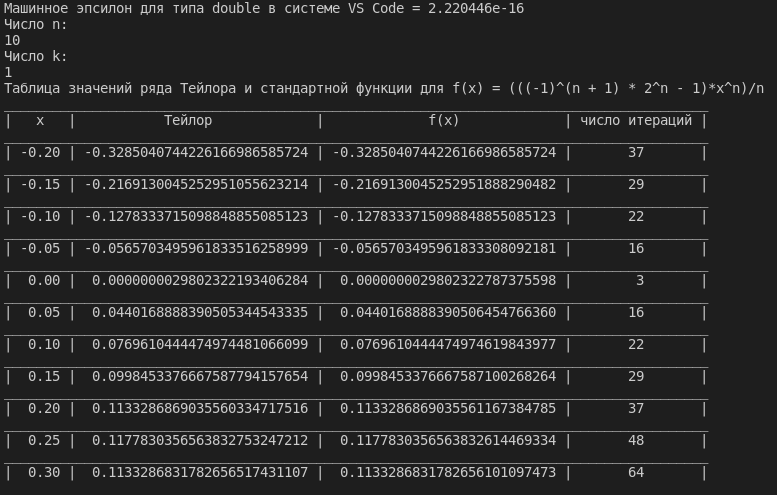
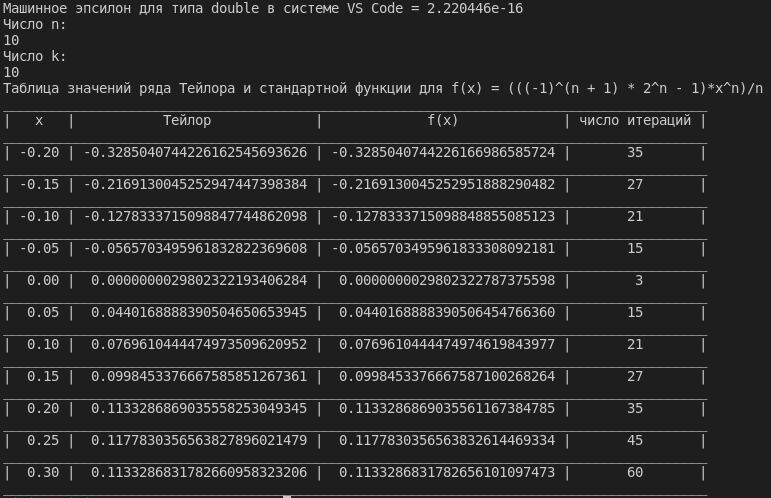
**Описание переменных**

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Тип** | **Назначение** |
| Epsillon | double | Машинный эпсилон |
| Taylor | Текущее значение ф-ии по Тейлору |
| a | Левая граница отрезка |
| b | Правая граница отрезка |
| x | Текущий аргумент ф-ии |
| function | Значение ф-ии |
| Buffer | Промежуточное хранение вычислений |
| n | int | Число разбиений отрезка |
| Counter | Счетчик |

**Проверка программы**

Входные данные – число n. В тестах n = 10. Также число k – в тестах 10 и 1

Выходные данные:

****

**Вывод**

В ходе работы я написал и протестировал программу на языке Си, которая оценивает точность вычисления функции с помощью ряда Тейлора. Основываясь на результатах, можно сделать вывод, что ряд Тейлора дает точные результаты, но существуют более быстрые способы представления трансцендентных функций.

**Исходный код**

#include <stdio.h>

#include <math.h>

int main(void) {

double Epsillon = 1.0;

double Taylor;

int n;

int k = 1000;

double a = -0.2f;

double b = 0.3f;

double x;

int Counter;

while (1.0 + (Epsillon / 2.0) > 1.0) {

Epsillon /= 2.0;

}

printf("Машинное эпсилон для типа double в системе VS Code = %e\n", Epsillon);

printf("Число n:\n");

scanf("%d", &n);

printf("Число k:\n");

scanf("%d", &k);

printf("Таблица значений ряда Тейлора и стандартной функции для f(x) = (((-1)^(n + 1) \* 2^n - 1)\*x^n)/n\n");

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

printf("| x | Тейлор | f(x) | число итераций |\n");

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

x = -0.2f;

for (int i = 0; i <= n; i++) {

double function;

function = log(1 + x - 2 \* pow(x,2));

Taylor = 0;

Counter = 1;

double Buffer = 1;

while (fabs(Buffer) > Epsillon \* k && Counter < 100) {

Buffer = ((pow(-1, Counter + 1) \* pow(2, Counter) - 1) \* pow(x, Counter)) / Counter;

Taylor += Buffer;

Counter += 1;

}

if (x < 0) {

if (Taylor < 0) {

if (Counter >= 10) {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

} else {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

}

} else {

if (Counter >= 10) {

if (function > 0) {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

} else {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

}

} else {

if (function > 0) {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

} else {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

}

}

}

} else {

if (Taylor < 0) {

if (Counter >= 10) {

if (function > 0) {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

} else {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

}

} else {

if (function > 0) {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

} else {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

}

}

} else {

if (Counter >= 10) {

if (function > 0) {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

} else {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

}

} else {

if (function > 0) {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

} else {

printf("| %.2f | %.25f | %.25f | %d |\n", x, Taylor, function, Counter);

}

}

}

}

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

x += (b - a) / n;

}

return 0;

}