Московский Авиационный Институт

(Национальный исследовательский университет)

**Факультет информационных технологий и прикладной математики**

**Кафедра №806 Вычислительная математика и программирование**

# Курсовой проект

**по курсам  
«Фундаментальная информатика», «Архитектура компьютера и информационных систем»  
I семестр**

**Задание 4**

**Процедуры и функции в качестве параметров**

Студент: Былькова К. А.

Группа: М8О-108Б-22

Номер по списку: 2

Руководитель: Сахарин Н. А.

Оценка: <…>

Дата: <…>

Подпись преподавателя:

**СОДЕРЖАНИЕ**

1. Задание.…….……………………………………………………………….3
2. Вариант……………………………………………………………………...3
3. Общий метод решения……………………………………………………..3
4. Общие сведения о программе……………………………………………..4
5. Функциональное назначение.……………………………………………..4
6. Описание логической структуры……………………………………….....4
7. Описание переменных, констант и подпрограмм………………………..5
8. Протокол…………………...………………………………………………..6
9. Входные данные……...…………………………………………………….9
10. Выходные данные…………………...……………………………………..9
11. Заключение………………………………………………………………...9

## Задание

Составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными способами (итераций, Ньютона и половинного деления - дихотомии). Нелинейные уравнения оформить как параметры-функции, разрешив относительно неизвестной величины в случае необходимости. Применить каждую процедуру к решению двух уравнений, заданных двумя строками таблицы, начиная с варианта с заданным номером. Если метод неприменим, дать математическое обоснование и графическую иллюстрацию.

## Вариант

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Уравнение | Отрезок, содержащий корень | Базовый метод | Приближенное значение корня |
| 2 | – 1 = 0 | [1, 2] | дихотомии | 1.0804 |
| 3 |  | [1, 1.5] | итераций | 1.1474 |

## Общий метод решения

Вычисление приближенного значений функций при помощи метода дихотомии, метода итераций и метода Ньютона.

Рассматривается уравнение вида *F(x)= 0*. Предполагается, что функция *F(x)* достаточно гладкая, монотонная на этом отрезке и существует единственный корень уравнения *x\* ∈ [a, b]*. на отрезке *[a, b]* ищется приближенное решение x с точностью *𝜀*, т.е. такое, что *|x - x\*| < 𝜀*.

Метод дихотомии - деление отрезка пополам с учётом того, что знак функции на концах отрезка должен быть разным: *F(a) \* F(b) < 0*. До тех пор, пока длина отрезка не будет меньше значения *𝜀*, процесс деления будет выполняться. Приближенное значение корня к моменту окончания итерационного процесса будет находиться примерно в середине заданного отрезка.

Метод итераций заключается в замене исходного уравнения *F(x) = 0* уравнением *f(x) = x*. Начальным приближенным значением корня является середина заданного отрезка *x(0) = (a + b)/2*. Итерационный процесс имеет вид: *x(k+1) = f(x(k))*. Процесс выполняется пока *|x (k+1) - x(k-1)| < 𝜀*

Метод Ньютона - частный случай метода итераций. Итерационный процесс представляет собой: *x(k+1) = x(k) – F(x(k))/F’(x(k))*.

## Общие сведения о программе

Аппаратное обеспечение: домашний ноутбук

Операционная система: Linux Ubuntu, версия 22.04.1 LTS

Язык и система программирования: C, GNU

Местонахождение файлов: /home/Kristina

Компиляция программы: g++ -lm cp3.c

Запуск программы: ./a.out

## Функциональное назначение

Программа предназначена для высокоточного вычисления вещественного значения трансцедентных функций в алгебраической форме с использованием ряда Тейлора и при помощи встроенных программных функций библиотеки языка Си.

## Описание логической структуры

Программа получает на вход заданный отрезок, находит значение уравнения *F(x) = 0* различными численными методами и выводит полученный корень уравнения.

## Описание переменных, констант и подпрограмм

Таблица 1. Описание функций программы

|  |  |  |
| --- | --- | --- |
| Функция | Входные аргументы | Описание |
| epsilon | double x | Функция для подсчета машинного ε |
| F2, F3 | double x | Вычисляет значение входной функции, подставляя значение x |
| F2\_x, F3\_x | double x | Функция, вычисляющая выраженный x |
| F2\_first\_derivative, F3\_first\_derivative | double x | Функция, вычисляющая первую производную от заданного уравнения |
| F2\_second\_derivative, F3\_second\_derivative | double x | Функция, вычисляющая вторую производную от заданного уравнения |
| F2\_x\_first\_derivative, F3\_x\_first\_derivative | double x | Функция, вычисляющая первую производную от уравнения, в котором выражен x |
| dichotomy | double F(double), double a, double b | Функция, вычисляющая значение уравнения F(x) = 0 методом дихотомии |
| iterations | double F\_x(double), double F\_x\_first\_derivative(double), double a, double b | Функция, вычисляющая значение уравнения F(x) = 0 методом итераций |
| newton | double F(double), double F\_first\_derivative(double), double F\_second\_derivative(double), double a, double b | Функция, вычисляющая значение уравнения F(x) = 0 методом Ньютона |

Таблица 2. Описание переменных

|  |  |
| --- | --- |
| Переменная | Значение |
| double abs\_eps | Абсолютный эпсилон |
| double relative\_eps | Относительный эпсилон |
| double a, b | Границы отрезка |
| long double x | Значение аргумента функции |

## Протокол

**Код программы:**

#include <stdio.h>

#include <math.h>

typedef double dbl;

dbl epsilon() {

dbl eps = 1.0;

while (1 + eps / 2.0 != 1)

eps /= 2.0;

return eps;

}

dbl F2(dbl x) {

return cos(x) - exp(-pow(x, 2) / 2) + x - 1;

}

dbl F2\_x(dbl x) {

return 1 + exp(-pow(x, 2) / 2) - cos(x);

}

dbl F2\_x\_first\_derivative(dbl x) {

return sin(x) - x \* exp(-pow(x, 2) / 2);

}

dbl F2\_first\_derivative(dbl x) {

return -sin(x) + x \* exp(-pow(x, 2) / 2) + 1;

}

dbl F2\_second\_derivative(dbl x) {

return -cos(x) + exp(-pow(x, 2) / 2) - pow(x,2) \* exp(-pow(x, 2) / 2);

}

dbl F3(dbl x) {

return 1 - x + sin(x) - log(1 + x);

}

dbl F3\_x(dbl x) {

return 1 + sin(x) - log(1 + x);

}

dbl F3\_x\_first\_derivative(dbl x) {

return cos(x) - 1/(1 + x);

}

dbl F3\_first\_derivative(dbl x) {

return cos(x) - 1 - 1/(1 + x);

}

dbl F3\_second\_derivative(dbl x) {

return 1/((1 + x) \* (1 + x)) - sin(x);

}

dbl dichotomy(dbl (\*F)(dbl), dbl a, dbl b, dbl relative\_eps, dbl abs\_eps) {

dbl x = (a + b) / 2;

if (F(a) \* F(b) < 0){

while (fabs(a - b) > fmax(relative\_eps \* fabs(x), abs\_eps)) {

x = (a + b) / 2;

if (F(x) \* F(a) < 0) {

b = x;

}

else {

a = x;

}

}

return x;

}

else {

return NAN;

}

}

dbl iterations(dbl (\*F\_x)(dbl), dbl (\*F\_x\_first\_derivative)(dbl), dbl a, dbl b, dbl relative\_eps, dbl abs\_eps) {

dbl x = (a + b) / 2;

if (fabs(F\_x\_first\_derivative(x)) < 1) {

while (fabs(F\_x(x) - x) >= fmax(relative\_eps \* fabs(x), abs\_eps)) {

x = F\_x(x);

}

return x;

}

else {

return NAN;

}

}

dbl newton(dbl (\*F)(dbl), dbl (\*F\_first\_derivative)(dbl), dbl (\*F\_second\_derivative)(dbl), dbl a, dbl b, dbl relative\_eps, dbl abs\_eps) {

dbl x = (a + b / 2);

if (fabs(F(x) \* F\_second\_derivative(x)) < (F\_first\_derivative(x) \* F\_first\_derivative(x))) {

while (fabs(F(x) / F\_first\_derivative(x)) > fmax(relative\_eps \* fabs(x), abs\_eps)) {

x -= F(x) / F\_first\_derivative(x);

}

return x;

}

else {

return NAN;

}

}

void result(dbl d, dbl i, dbl n) {

if (d != NAN) printf("The root obtained by the dichotomy method: %.10f\n", d);

else printf("The dechotomy method isn't suitable\n");

if (i != NAN) printf("The root obtained by the iterations method: %.10f\n", i);

else printf("The iterations method isn't suitable\n");

if (n != NAN) printf("The root obtained by the Newton's method: %.10f\n", n);

else printf("The Newton's method isn't suitable\n");

}

int main() {

dbl abs\_eps = epsilon();

dbl relative\_eps = sqrt(abs\_eps);

dbl a2 = 1, b2 = 2;

dbl a3 = 1, b3 = 1.5;

dbl d1 = dichotomy(F2, a2, b2, relative\_eps, abs\_eps);

dbl i1 = iterations(F2\_x, F2\_x\_first\_derivative, a2, b2, relative\_eps, abs\_eps);

dbl n1 = newton(F2, F2\_first\_derivative, F2\_second\_derivative, a2, b2, relative\_eps, abs\_eps);

printf("Machine epsilon for long double = %.16e\n", abs\_eps);

printf("Function cos(x) - exp(-0.5\*x^2) + x - 1\n");

result(d1, i1, n1);

printf("\n");

dbl d2 = dichotomy(F3, a3, b3, relative\_eps, abs\_eps);

dbl i2 = iterations(F3\_x, F3\_x\_first\_derivative, a3, b3, relative\_eps, abs\_eps);

dbl n2 =newton(F3, F3\_first\_derivative, F3\_second\_derivative, a3, b3, relative\_eps, abs\_eps);

printf("Function 1 - x + sin(x) - ln(1 + x)\n");

result(d2, i2, n2);

return 0;

}

## Входные данные

Отсутствуют.

## Выходные данные

kristina@kristina-VirtualBox:~/Рабочий стол/course-w$ g++ -lm cp4.c && ./a.out

Machine epsilon for long double = 2.2204460492503131e-16

Function cos(x) - exp(-0.5\*x^2) + x - 1

The root obtained by the dichotomy method: 1.0894428008

The root obtained by the iterations method: 1.0894428008

The root obtained by the Newton's method: 1.0894428008

Function 1 - x + sin(x) - ln(1 + x)

The root obtained by the dichotomy method: 1.1474388506

The root obtained by the iterations method: 1.1474388506

The root obtained by the Newton's method: 1.1474388506

## Заключение

В результате выполнения данного курсового проекта были получены навыки работы с процедурами и функциями в качестве параметров. Было изучено вычисление машинного эпсилона и различных численных методов, таких как: метод дихотомии (половинного деления), метод итераций и метод Ньютона. Оценивая полученные данные, можно сказать, что каждый из методов неидеален, так как для поиска корня необходимо знать точные границы отрезка. Также при увеличении точности вычислений затраты по времени быстро увеличиваются.

Благодаря использованию универсальных функций, которые принимают в качестве аргументов указатели на другие функции, удалось избежать дублирования кода.