

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский Авиационный Институт»  
(Национальный Исследовательский Университет)

Институт: №8 «Компьютерные науки и прикладная математика»  
Кафедра: 806 «Вычислительная математика и программирование»

Курсовая работа  
по курсу «Фундаментальная  
информатика»  
I семестр  
Задание 4  
«Процедуры и функции в качестве параметров»

Группа	М8О-109Б-22
Студент	Филиппов А. М.
Преподаватель	Сысоев М. А.
Оценка	
Дата	26 апреля 2022 г.

Москва, 2022

## Задание

Составить программу на Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (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, например, с использованием gnuplot.

### Варианты 9, 10:

№	Уравнение	Отрезок, содержащий корень	Базовый метод	Приближенное значение корня
9	$x^2 - \ln(1+x) - 3 = 0$	[2, 3]	итераций	2.0267
10	$2x \cdot \sin x - \cos x = 0$	[0.4, 1]	Ньютона	0.6533

# Теоретическая часть

## 1. Метод итераций

Идея заключается в замене исходного уравнения  $f(x) = 0$  на уравнение  $x = f(x)$ . Перед началом итерационного процесса необходимо проверить условие сходимости:  $|f'(x)| < 1$ ,  $x \in [a, b]$ . Изначально  $x$  равен  $(a + b)/2$ . Итерационный процесс:  $x_{i+1} = f(x_i)$ .

## 2. Метод Ньютона

Является частным случаем метода итераций, отличается условие выхода из цикла:  $|f(x) * f''(x)| < (f'(x))^2$  и итерационным процессом:  $x_{i+1} = x_i - (f(x_i)/f'(x_i))$ .

## Алгоритм решения

Для начала с помощью функций `IsNewtonConvergent` и `IsIterationsConvergent` проверим методы на сходимость. Для каждого сходящегося метода с его помощью вычислим корень соответствующего уравнения. Затем подставим корни в уравнение для проверки.

## Использованные в программе переменные

Название переменной	Тип переменной	Смысл переменной
a	long double	Начало отрезка
b	long double	Конец отрезка
x	long double	Значения в промежутке [a;b], для которого вычисляются значения
step	long double	Значение, прибавляемое к x на каждом шаге
root	long double	Корень уравнения
result	long double	Значение уравнения после подстановки корня
LDBL_EPSILON	long double	Машинный эпсилон. Для long double $\varepsilon = 1.08 * 10^{-19}$

## Исходный код программы:

```
#include <stdio.h>
#include <float.h>
#include <math.h>

long double f9(long double x) {
    return sqrtl(logl(1 + x) + 3);          //  $x^2 - \log(1 + x) - 3 = 0$ 
}

long double f9_der(long double x) {
    return 1 / (2 * (1 + x) * sqrtl(logl(1 + x) + 3));
}

int IsIterationConvergent(
    long double (*der)(long double),
    long double a,
    long double b) {
    int res = 1;
    long double step = (b - a) / 100;
    for (long double x = a; x <= b; x += step) {
        if (der(x) >= 1) {
            res = 0;
        }
    }
    return res;
}

long double IterationsMethod(
    long double (*f)(long double),
    long double a,
    long double b) {
    long double x0 = (a + b) / 2;
    long double x1 = f(x0);
    while (fabsl(x1 - x0) >= LDL_EPSILON) {
        x0 = x1;
        x1 = f(x0);
    }
    return x1;
}

long double f10(long double x) {
    return 2 * x * sinl(x) - cosl(x);      //  $2 * x * \sin(x) - \cos(x)$ 
    = 0
}

long double f10_der1(long double x) {
    return 2 * x * cosl(x) + 3 * sinl(x);
}

long double f10_der2(long double x) {
    return 5 * cosl(x) - 2 * x * sinl(x);
}

int IsNewtonConvergent(
    long double (*f)(long double),
    long double (*der1)(long double),
    long double (*der2)(long double),
    long double a,
    long double b) {
    int res = 1;
    long double step = (b - a) / 100;
    for (long double x = a; x <= b; x += step) {
        if (fabsl(f(x) * der2(x)) >= powl(der1(x), 2)) {
```

```

        res = 0;
    }
}
return res;
}

long double NewtonMethod(
    long double (*f)(long double),
    long double (*der)(long double),
    long double a,
    long double b) {
    long double x0 = (a + b) / 2;
    long double x1 = x0 - f(x0) / der(x0);
    while (fabs1(x1 - x0) >= LDBL_EPSILON) {
        x0 = x1;
        x1 = x0 - f(x0) / der(x0);
    }
    return x1;
}

int main() {
    long double a = 2, b = 3;
    printf("Function 9 (Iterations Method):\n\t\tx^2 - log1(1 + x) - 3 = 0\n");
    if (IsIterationConvergent(f9, der, a, b)) {
        printf("Convergence check:\tOK!\n");

        long double root = IterationsMethod(f9, a, b);
        printf("Approximated root:\t%.10Lf\n", root);

        long double result = root * root - log1(1 + root) - 3;
        printf("Result of inserting root:\t%.10Lf\n\n", result);
    } else {
        printf("Convergence check:\t FAIL!\n\n");
    }

    a = 0.4;
    b = 1;
    printf("Function 10 (Newton Method):\n\t\t2 * x * sin1(x) - cos1(x) = 0\n");
    if (IsNewtonConvergent(f10, f10_der1, f10_der2, a, b)) {
        printf("Convergence check:\tOK!\n");

        long double root = NewtonMethod(f10, f10_der1, a, b);
        printf("Approximated root:\t%.10Lf\n", root);

        long double result = f10(root);
        printf("Result of inserting root:\t%.10Lf\n", result);
    } else {
        printf("Convergence check:\t FAIL!\n\n");
    }
}

```

## Входные данные

Нет

## Выходные данные

Программа должна вывести для каждого уравнения сходится метод или нет.

Если метод сходится, вывести приближенный корень уравнения, а затем вывести значение уравнения, в который будет подставлен корень.

## Протокол исполнения и тесты

### Тест №1 (корни совпали)

Входные данные:

Выходные данные:

```
Function 9 (Iterations Method):
      x^2 - logl(1 + x) - 3 = 0
Convergence check:      OK!
Approximated root:      2.0266892632
Result of inserting root:      -0.0000000000

Function 10 (Newton Method):
      2 * x * sinl(x) - cosl(x) = 0
Convergence check:      OK!
Approximated root:      0.6532711871
Result of inserting root:      -0.0000000000

Process finished with exit code 0
```

## Вывод

Недостатком почти всех методов нахождения корней является то, что при них позволяют найти лишь один корень функции, к тому же, мы не знаем какой именно. Чтобы найти другие корни, можно было бы брать новые стартовые точки и применять метод вновь, но нет гарантии, что при этом итерации сойдутся к новому корню, а не к уже найденному, если вообще сойдутся.