Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский авиационный институт (национальный исследовательский университет)»

НАПРАВЛЕНИЕ «МАТЕМАТИКА И ИНФОРМАТИКА»

Курсовая работа по информатике

На тему: «Процедуры и функции в качестве параметров»

Работу выполнил
студент I курса
очного отделения
группы 80-104Б
Железнов И.В.
Преподаватель:
Аспирант каф.806
Потенко М.А.

Москва 2022

Содержание:

Введение

Цели КП: Изучение процедур и функций, способов их реализации на Си, а также изучение различных численных методов: метод дихотомии (метод половинного деления), метод итераций, метод Ньютона.

Задача КП: составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами. Применить каждый метод к решению двух уравнений, заданных таблицей. Если метод неприменим, дать математическое обоснование и графическую иллюстрацию.

Уравнение	Вариант	Отрезок, содержащий корень	Приближенное значение корня
$\tan\left(\frac{x}{2}\right)-\cot\left(\frac{x}{2}\right)+x=0$	14	[1;2]	1.0769
$0.4+\arctan\left(\sqrt{x}\right)-x=0$	15	[1;2]	1.2388

Основная часть

Прежде чем составлять программу на языке Си, необходимо подробно изучить каждый численный метод и проверить, возможно ли решить предложенное уравнение данным методам. Опишу каждый численный метод подробнее:

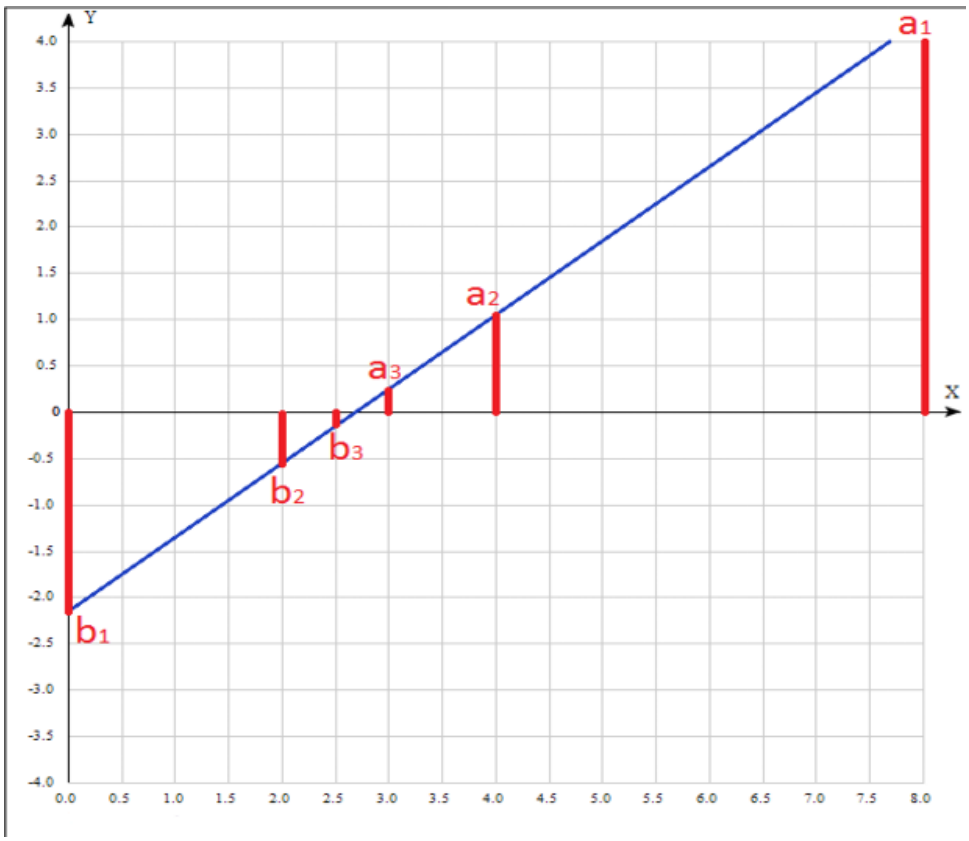
Метод половинного деления

Пусть корень уравнения f(x)=0 отделен на отрезке [a;b], то есть на этом отрезке имеется единственный корень, а функция на данном отрезке непрерывна.

Метод половинного деления позволяет получить последовательность вложенных друг в друга отрезков $[a_1;b_1]$, $[a_2;b_2]$, …,$[a_3;b_3]$,…, $[a_n;b_n]$, таких что $f(a_i).f(b_i) < 0$, где i=1,2,…,n, а длина каждого последующего отрезка вдвое меньше длины предыдущего.

Последовательное сужение отрезка вокруг неизвестного значения корня обеспечивает выполнение на некотором шаге n неравенства $|b_n - a_n| < Eps$.

Вот так выглядит метод, если изобразить его графически:



Как видно b_n и a_n с каждой итерацией становятся все ближе и ближе к искомому значению.

Проверю значения данных уравнений на концах отрезках:

Вариант 14: f(1) = -0.2841; f(2) = 2.9153;

Вариант 15: f(1) = 0.1853; f(2) = -0.6446;

Как видно, метод применим, так как на концах отрезка значения функций принимают разные знаки.

Метод итераций

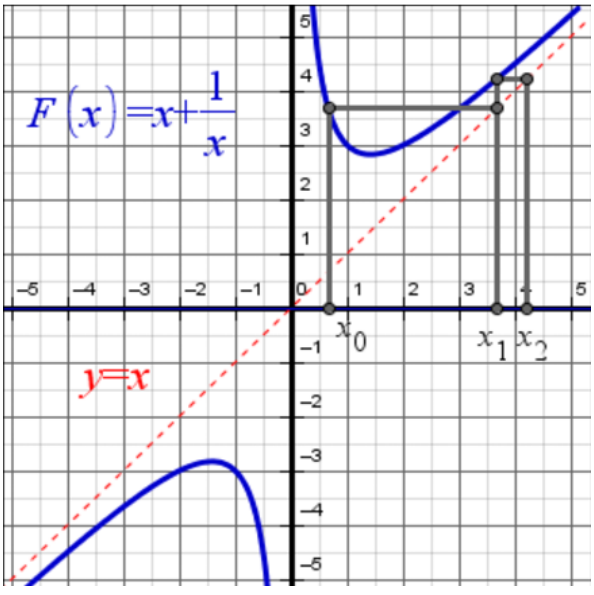
Данный метод называют также методом последовательных приближений, методом повторных подстановок, методом простых итераций и т.п.

Пусть дано f(x) = 0 (1)

Будем вместо уравнения (1) рассматривать равносильное ему уравнение
x = F(x), (2)
где F(x) = f(x) + x.

Пустьx_0 – произвольное число (начальное приближение искомого корня уравнения (1)).
Рассмотрим последовательность
$x_1 = F(x_0)$, $x_2 = F(x_1)$, …, $x_n = F(x_{n-1})$, …
Если эта последовательность имеет предел, то он и есть решение (корень) уравнения (2), а значит, и уравнения (1).

Наглядно процесс показан на рисунке ниже.



Не при всех условиях итерационный процесс сходится к корню уравнения x. Для того чтобы итерационный процесс был сходящимся, необходимо в окрестности корня выполнение следующего неравенства:

|F’(x)| < 1;

Найдем подходящее уравнение F(x) для приведенных выше вариантов:

Приведение уравнений

Вариант 14:

$$\tan\left(\frac{x}{2}\right)-\cot\left(\frac{x}{2}\right)+x=0;$$

$$\frac{\sin(x)}{\cos(x)}-\frac{\cos(x)}{\sin(x)}+x=0;$$

$$\frac{\sin^2\left(\frac{x}{2}\right)-\cos^2\left(\frac{x}{2}\right)}{\sin\left(\frac{x}{2}\right)\cos\left(\frac{x}{2}\right)}+x=0;$$

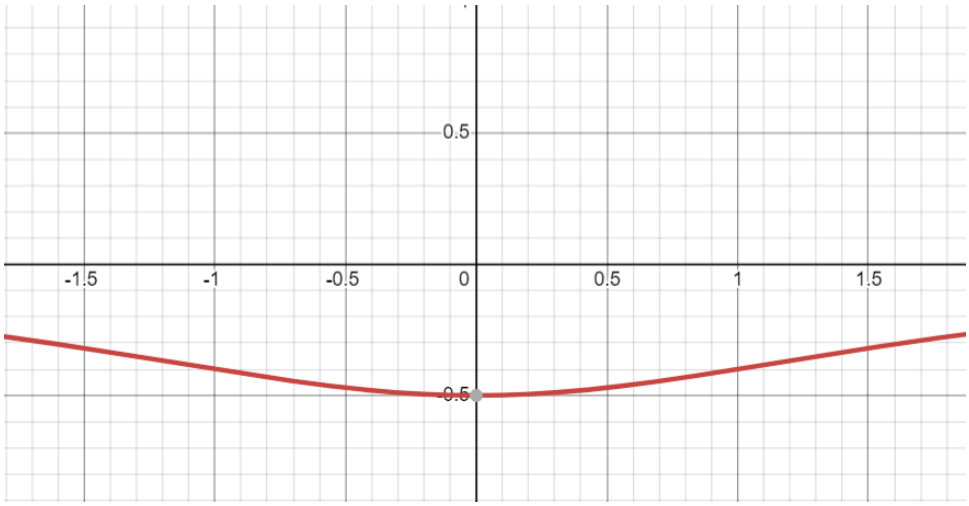
$$-2\,ctg(x)+x=0;$$

$$x=arcctg\left(\frac{x}{2}\right);$$

arcctg имеет область значений от 0 до π, этому интервалу принадлежит наш корень.

Посмотрим на график производной:

$$arcctg'\left(\frac{x}{2}\right)=\frac{-2}{x^2+4}$$



Как видно из графика (касный цвет), представленного ниже, производная удовлетворяет условию | F’(x)| < 1, значит мы можем воспользоваться данной функцией.

Вариант 15:

$$0.4+\arctan\left(\sqrt{x}\right)-x=0;$$

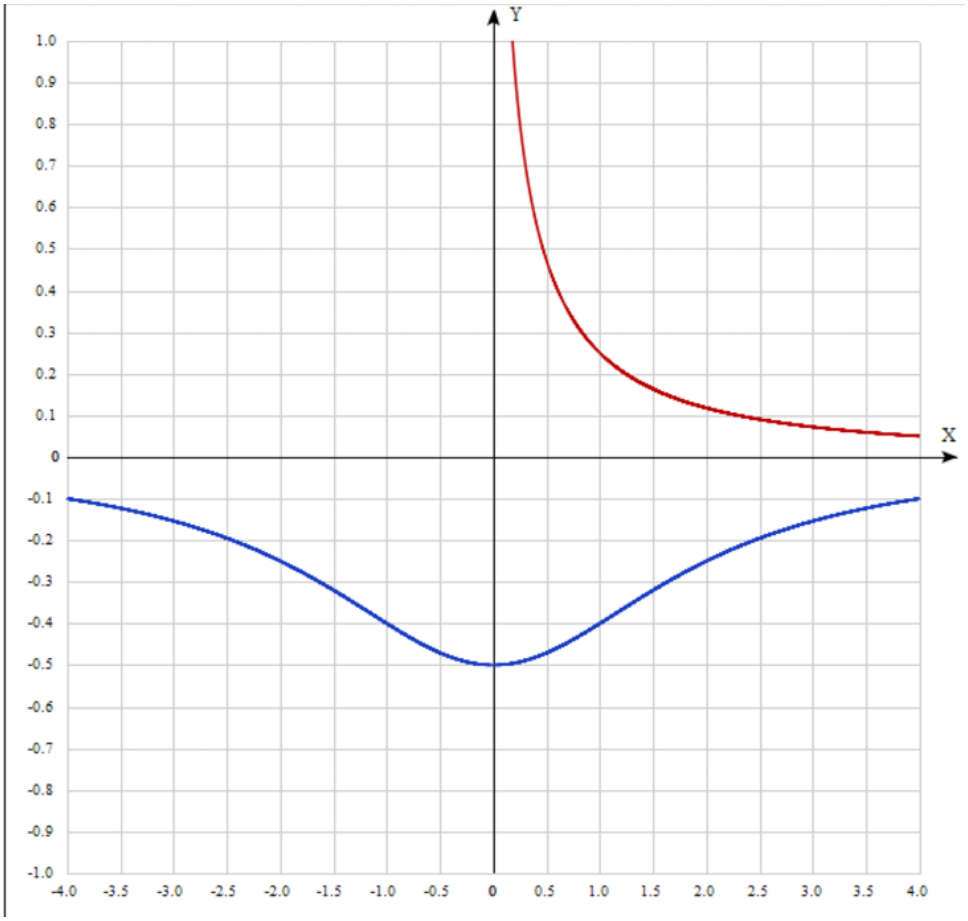
$$x=0.4+atan\left(\sqrt{x}\right);$$

arcctg имеет область значений – π/2 от до π/2. Области значений F(x) принадлежит наш корень.

Посмотрим на график производной:

$$F'(x)=\frac{1}{(2(x+1)\sqrt{x})}$$

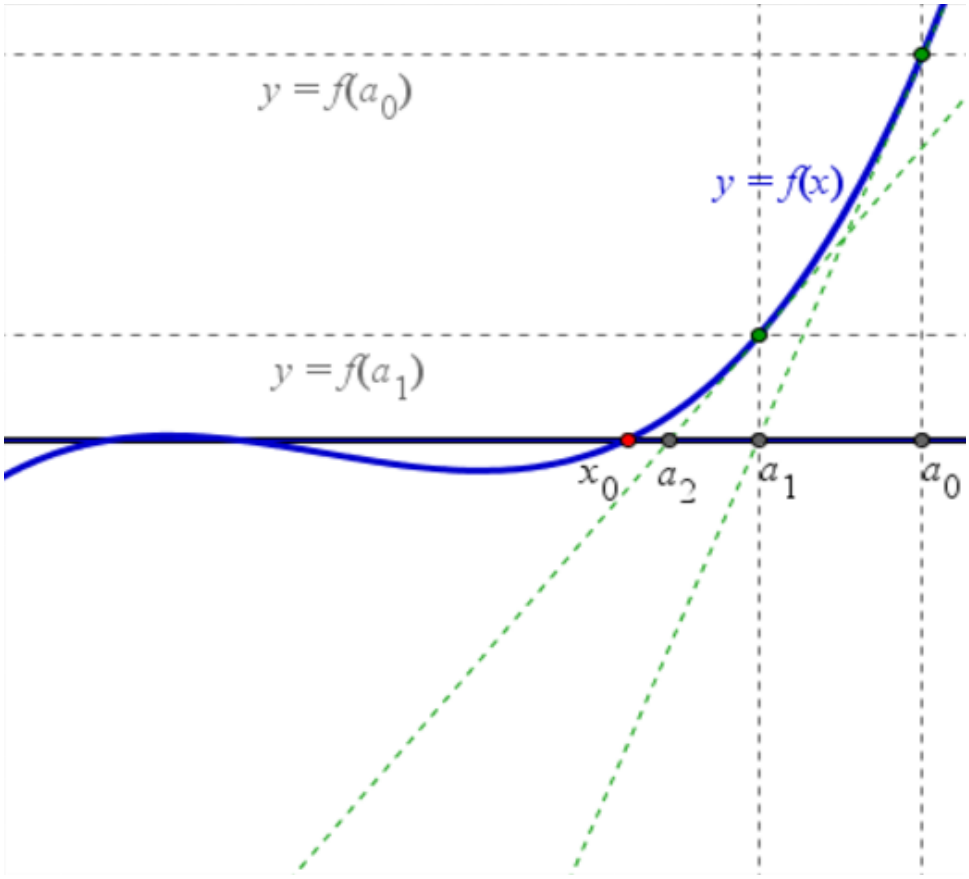
Как видно из графика (красный цвет), представленного ниже, производная удовлетворяет условию | F’(x)| < 1, значит мы можем воспользоваться данной функцией.



Метод Ньютона

Суть метода можно сформулировать так.

Берется какое-либо число a_1 как можно ближе к искомому корню x_0 и принимается за первое приближение корня (см. рис.). Затем через точку A_1 с координатами $(a_1; f(a_1))$ проводится касательная к графику функции y = f(x) до пересечения с осью абсцисс в точке $(a_2; 0)$. Эта точка пересечения дает нам второе приближение корня x_0. Повторяя этот процесс, получаем все более и более точные значения $a_0; a_1; a_2; \ldots$ корня x_0.



С помощью уравнения касательной можно вывести рекуррентную формулу, выражающую очередное, i-е приближение через предыдущее:

$$x_{k+1}=x_k-\frac{f_k(x)}{f_k'(x)}$$

Условие сходимости:

$|f(x)f''(x)|<(f'(x))^2$ на [a,b];

Проверим, подходит ли метод Ньютона для решения представленных уравнений:

Вариант 14:

$$f(x)=tg(\frac{x}{2})-ctg(\frac{x}{2})+x;$$

$$f'(x)=\frac{1}{2}\cos^{-2}(\frac{x}{2})+\frac{1}{2}\sin^{-2}(\frac{x}{2})+1;$$

$$f''(x)=\frac{1}{2}(\cos^{-3}(\frac{x}{2})\sin(\frac{x}{2})-\sin^{-3}(\frac{x}{2})\cos(\frac{x}{2}));$$

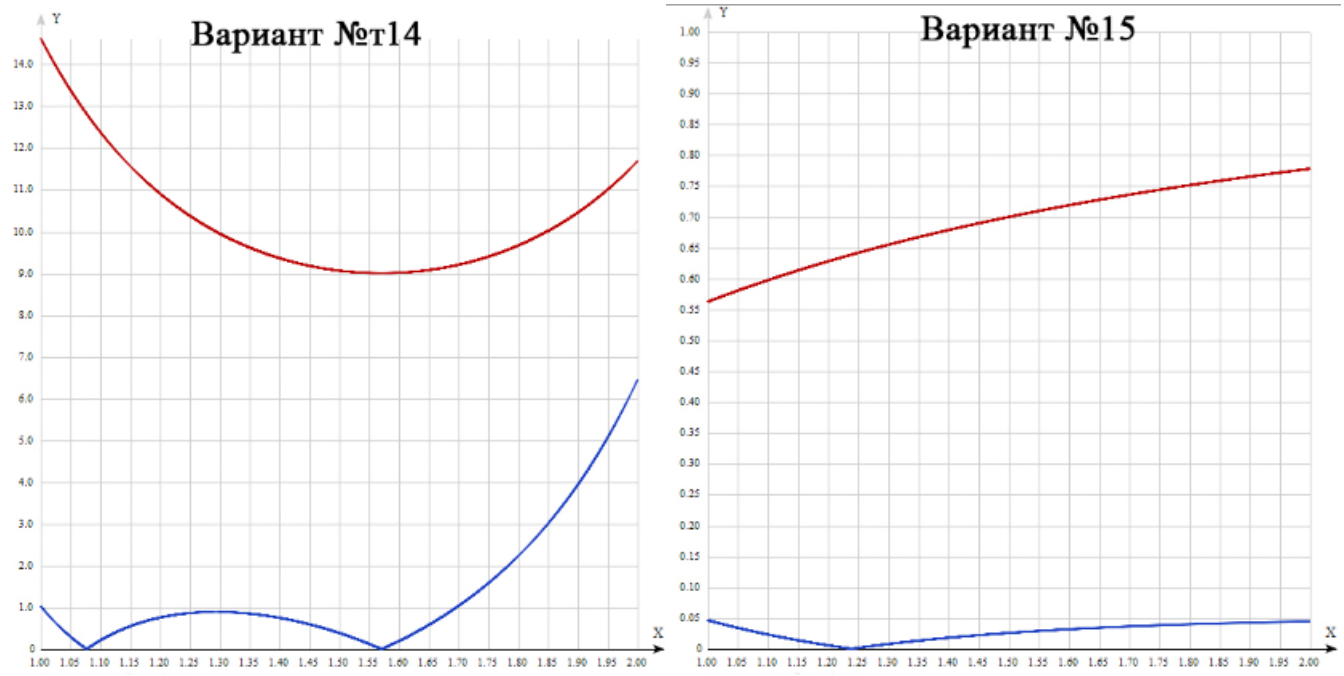
Вариант 15:

$$f(x)=0.4+arctg(\sqrt{x})-x;$$

$$f'(x)=\frac{1}{2(x+1)\sqrt{x}}-1;$$

$$f''(x)=-\frac{3x+1}{4(x^3+2x^2+x)\sqrt{x}};$$

Построим графики функций $g(x)=|f(x)f''(x)|$ (синий цвет) и $q(x)=(f'(x))^2$ (красный цвет).



Как видно из графиков функций, условие сходимости выполнено на данных отрезках [a;b], значит мы можем использовать данный метод для решения уравнений.

Зная, что каждый из методов мы можем применить для решения каждого уравнения, составим программу на Си, вычисляющую их корни.

Составим таблицу использованных функций и их переменных:

Функция	Переменные функции	Что делает
—	Double eps – значение эпсилон, глобальная переменная	Хранит в себе значение машинного эпсилона

Double epsylone()	Double epsylone – значение эпсилон	Вычисляет значение машинного эпсилон
Double a(double x)	Double x	Нужна для вычисление правильного порядка машинного эпсилон
Double f1(double x)	Double x – значение точки, для которой нужно вычислить f1(x)	Вычисляет значение функции из варианта 14 в данной точке
Double f2(double x)	Double x (аналогично с функцией f1)	Аналогично с функцией f1, но для функции из варианта 15
Double F1(double x)	Double x – значение точки, для которой нужно вычислить F1(x)	Вычисляет значение дополнительной функции из варианта 14 в данной точке
Double F2(double x)	Double x (аналогично с функцией F1)	Аналогично с функцией F1, но для функции из варианта 15
Double derivative_f1(double x)	Double x – значение точки, для которой нужно вычислить derivative_f1(x)	Вычисляет значение производной функции из варианта 14 в данной точке
Double derivative_f2(double x)	Double x (аналогично с derivative_f1)	Аналогично с функцией derivative_f1, но для функции из варианта 15
double dichotomy_method(double a, double b, double (*f) (double))	double a, b – значение правого и левого концов отрезка соответственно; double (*f) (double) – указатель на функцию, для которой вычисляется корень; int i – количество итераций; eps – значение эпсилон;	Вычисляет корень уравнения методом дихотомии
double iterative_methode(double x1, double x2, double (*F) (double))	double x1, double x1 – нужны для вычисления первого приближения, а затем последующих; double (*F) (double) — указатель на дополнительную функцию; int i – количество итераций; eps – значение эпсилон;	Вычисляет корень уравнения методом итераций
double Newton_method(double x1, double x2, double (*f) (double), double (*derivative_f)(double))	double x1, double x1 – нужны для вычисления первого приближения, а затем последующих; double (*f) (double) — указатель на функцию, для которой вычисляется корень; double (*derivative_f) (double)) – указатель на функцию, которая вычисляет производную; int i – количество итераций; eps – значение эпсилон;	Вычисляет корень уравнения методом Ньютона
void print_table(double a, double b, double (*f) (double), double (*F) (double), double (*derivative_f) (double))	double a - значение левого конца отрезка; double b – значение правого конца отрезка; Double (*f) (double) — указатель на функцию f(x); double (*F) (double) — указатель на дополнительную функцию f(x); double (*derivative_f) (double) — указатель на производную функции f(x);	Для конкретной функции печатает таблицу, которая состоит из корня функции на [a;b], метода, которым он вычислен, и количества итераций, ушедших на вычисление корня данным методом.

void main()	Int k –порядок эпсилон; double a, b – значение правого и левого концов отрезка соответственно; eps – глобальная переменная, которой присваивается значение эпсилон	Вычисляет нужный порядок эпсилон, вызывает функцию print_table для уравнения 14 и 15.

Код программы

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

double eps; // Глобальная переменная

double get_machine_eps()
{
   double epsylone = 1.0;
   while((1.0 + epsylone / 2) > 1.0) {
      epsylone /= 2;
   }
   return epsylone;
}

double f1(double x)
{
return tan(x / 2) - (1 / tan(x / 2)) + x;
}

double F1(double x)
{
return M_PI / 2 - atan(x / 2);
}

double derivative_f1(double x)
{
return (1 / pow(cos(x / 2), 2) + 1 / pow(sin(x / 2), 2) + 2) / 2;
```

```
}

double f2(double x)
{
return 0.4 + atan(sqrt(x)) - x;
}

double F2(double x)
{
return 0.4 + atan(sqrt(x));
}

double derivative_f2(double x)
{
return (1 / (2 * (pow(x, 3 / 2) + pow(x, 1 / 2)))) - 1;
}

void dichotomy_method(double a, double b, double (*f) (double))
{
double c;
int i = 0;

while(fabs(a - b) >= eps)
{
c = (a + b) / 2;
if ((*f)(a) * (*f)(c) > 0) a = c;
else b = c;
i++;
}
printf("| Dichotomy method  | %4d | %.16lf |\n", i, (a + b) / 2);
}

void iterative_method(double x1, double x2, double (*F) (double))
```

```
{
int i = 0;

x2 = (x1 + x2) / 2;
do {
x1 = x2;
x2 = (*F)(x1);
i++;
} while(fabs(x1 - x2) >= eps);
printf("| Iterative methode | % 4d | %.16lf |\n", i, x2);
}

void newton_method(double x1, double x2, double (*f) (double), double (*derivative_f) (double))
{
int i = 0;

x2 = (x1 + x2) / 2;
do {
x1 = x2;
x2 = x1 - (*f)(x1) / (*derivative_f)(x1);
i++;
} while(fabs(x1 - x2) >= eps);
printf("| Newton's method   | %4d | %.16lf |\n", i, x2);
}

void print_table(double a, double b, double (*f) (double), double (*F) (double), double (*derivative_f)
(double))
{
printf(" _____\n");
printf("|    Method name    | iter |        root        |\n");
printf("|———————————————————————|——————|————————————————————
————|\n");
dichotomy_method(a, b, f);
```

```
printf("|————————————————————————————————————————|————————————|————————————————————————————————————————————|\n");
iterative_method(a, b, F);
printf("|————————————————————————————————————————|————————————|————————————————————————————————————————————|\n");
newton_method(a, b, f, derivative_f);
printf(" ——————————————————————————————————————————————————————————————————————————————————————————————————————\n");
}

void main()
{
double a = 1.0, b = 2.0;
int k;

scanf("%d", &k);
eps = get_machine_eps() * pow(10, k);
printf("epsylone = %g\n\n", eps);

printf("          Root for equation №1\n");
print_table(a, b, f1, F1, derivative_f1);
printf("          Root for equation №2\n");
print_table(a, b, f2, F2, derivative_f2);
}
```

Тесты программы

```
keinpop@DESKTOP-T6SLHUS:/mnt/c/labs$ gcc kp4.c -lm
keinpop@DESKTOP-T6SLHUS:/mnt/c/labs$ ./a.out
5
epsylone = 2.22045e-11

            Root for equation №1
 ---------------------------------------------
|    Method name     | iter |        root        |
|--------------------|------|--------------------|
| Dichotomy method   |   36 | 1.0768739863051451 |
|--------------------|------|--------------------|
| Iterative methode  |   27 | 1.0768739863087642 |
|--------------------|------|--------------------|
| Newton's method    |    5 | 1.0768739863118038 |
 ---------------------------------------------
            Root for equation №2
 ---------------------------------------------
|    Method name     | iter |        root        |
|--------------------|------|--------------------|
| Dichotomy method   |   36 | 1.2388399775736616 |
|--------------------|------|--------------------|
| Iterative methode  |   16 | 1.2388399775757328 |
|--------------------|------|--------------------|
| Newton's method    |    8 | 1.2388399775742611 |
 ---------------------------------------------
```

```
keinpop@DESKTOP-T6SLHUS:/mnt/c/labs$ ./a.out
10
epsylone = 2.22045e-06

            Root for equation №1
 ---------------------------------------------
|    Method name     | iter |        root        |
|--------------------|------|--------------------|
| Dichotomy method   |   19 | 1.0768747329711914 |
|--------------------|------|--------------------|
| Iterative methode  |   15 | 1.0768737221456464 |
|--------------------|------|--------------------|
| Newton's method    |    4 | 1.0768739863118031 |
 ---------------------------------------------
            Root for equation №2
 ---------------------------------------------
|    Method name     | iter |        root        |
|--------------------|------|--------------------|
| Dichotomy method   |   19 | 1.2388391494750977 |
|--------------------|------|--------------------|
| Iterative methode  |    9 | 1.2388400986475925 |
|--------------------|------|--------------------|
| Newton's method    |    5 | 1.2388399730130997 |
 ---------------------------------------------
```

```
keinpop@DESKTOP-T6SLHUS:/mnt/c/labs$ ./a.out
1
epsylone = 2.22045e-15

            Root for equation №1
 ---------------------------------------------
|    Method name     | iter |        root        |
|--------------------|------|--------------------|
| Dichotomy method   |   49 | 1.0768739863118038 |
|--------------------|------|--------------------|
| Iterative methode  |   37 | 1.0768739863118033 |
|--------------------|------|--------------------|
| Newton's method    |    5 | 1.0768739863118038 |
 ---------------------------------------------
            Root for equation №2
 ---------------------------------------------
|    Method name     | iter |        root        |
|--------------------|------|--------------------|
| Dichotomy method   |   49 | 1.2388399775741474 |
|--------------------|------|--------------------|
| Iterative methode  |   21 | 1.2388399775741481 |
|--------------------|------|--------------------|
| Newton's method    |   11 | 1.2388399775741474 |
 ---------------------------------------------
```

```
keinpop@DESKTOP-T6SLHUS:/mnt/c/labs$ ./a.out
4
epsylone = 2.22045e-12

            Root for equation №1
 ---------------------------------------------
|    Method name     | iter |        root        |
|--------------------|------|--------------------|
| Dichotomy method   |   39 | 1.0768739863115115 |
|--------------------|------|--------------------|
| Iterative methode  |   29 | 1.0768739863113470 |
|--------------------|------|--------------------|
| Newton's method    |    5 | 1.0768739863118038 |
 ---------------------------------------------
            Root for equation №2
 ---------------------------------------------
|    Method name     | iter |        root        |
|--------------------|------|--------------------|
| Dichotomy method   |   39 | 1.2388399775745711 |
|--------------------|------|--------------------|
| Iterative methode  |   17 | 1.2388399775744656 |
|--------------------|------|--------------------|
| Newton's method    |    9 | 1.2388399775741445 |
 ---------------------------------------------
```

```
keinpop@DESKTOP-T6SLHUS:/mnt/c/labs$ ./a.out
8
epsylone = 2.22045e-08

             Root for equation №1
 ---------------------------------------------
|    Method name     | iter |        root         |
|--------------------|------|---------------------|
| Dichotomy method   |   26 | 1.0768739804625511  |
|--------------------|------|---------------------|
| Iterative methode  |   19 | 1.0768739803481329  |
|--------------------|------|---------------------|
| Newton's method    |    5 | 1.0768739863118038  |
 ---------------------------------------------

             Root for equation №2
 ---------------------------------------------
|    Method name     | iter |        root         |
|--------------------|------|---------------------|
| Dichotomy method   |   26 | 1.2388399764895439  |
|--------------------|------|---------------------|
| Iterative methode  |   11 | 1.2388399824486194  |
|--------------------|------|---------------------|
| Newton's method    |    6 | 1.2388399777073364  |
 ---------------------------------------------
```

```
keinpop@DESKTOP-T6SLHUS:/mnt/c/labs$ ./a.out
11
epsylone = 2.22045e-05

             Root for equation №1
 ---------------------------------------------
|    Method name     | iter |        root         |
|--------------------|------|---------------------|
| Dichotomy method   |   16 | 1.0768814086914062  |
|--------------------|------|---------------------|
| Iterative methode  |   12 | 1.0768785220726500  |
|--------------------|------|---------------------|
| Newton's method    |    4 | 1.0768739863118031  |
 ---------------------------------------------

             Root for equation №2
 ---------------------------------------------
|    Method name     | iter |        root         |
|--------------------|------|---------------------|
| Dichotomy method   |   16 | 1.2388381958007812  |
|--------------------|------|---------------------|
| Iterative methode  |    7 | 1.2388429848334388  |
|--------------------|------|---------------------|
| Newton's method    |    4 | 1.2388401337669173  |
 ---------------------------------------------
```

Заключение

В ходе выполнения курсового проекта были изучены такие вычислительные методы, как метод дихотомии, метод итераций, метод Ньютона. В ходе проведения тестирования было выявлено, что метод Ньютона наиболее эффективен, он вычисляет корень уравнения за наименьшее количество итераций, а метод половинного деления, наоборот самый медленный, однако он наиболее прост в реализации.

Так же во время выполнения я изучил дополнительные возможности языка Си - использование указателей на функции. Это тема 2-го семестра обучения. Я думаю, эти знания помогут мне в будущем разобраться в этом более ясно и удобно.

Список литературы

1. http://www.machinelearning.ru/wiki/index.php?title=Методы_дихотомии
2. https://ru.wikipedia.org/wiki/
3. https://ru.wikipedia.org/wiki
4. https://metanit.com/cpp/c/5.11.php
5. https://learnc.info/c/function_pointers.html
6. https://www.desmos.com/calculator?lang=ru