1. **NUMERIK HATALAR ve BILGISAYAR ARITMETIGI**

**ODEV 1 - 18.02.2020:**

ex  fonksiyonunda e0.5 =1.648721271’dir. Bu sayiya 4 basamakli bir sistemde kesme ve yuvarlama yapan C kodunu yaziniz.

#include <stdio.h>

#include <math.h>

double kesme(double normalize, int basamaksistemi)

{

    int tmp = normalize \* pow(10, basamaksistemi);

    normalize = tmp / (pow(10, basamaksistemi) \* 1.0);

    return normalize;

}

double yuvarla(double normalize, int basamaksistemi)

{

    int yuvarlamabasamagi;

    int tmp = normalize \* pow(10, basamaksistemi + 1);

    yuvarlamabasamagi = tmp % 10;

    tmp /= 10;

    if (yuvarlamabasamagi >= 5)

    {

        tmp++;

    }

    normalize = tmp / (pow(10, basamaksistemi) \* 1.0);

    return normalize;

}

int main(int argc, char const \*argv[])

{

    int kuvvet = 0;

    double sayi = exp(0.5);

    double normalize;

    int basamaksistemi;

    printf("basamak sistemini giriniz. \n");

    scanf("%d", &basamaksistemi);

    normalize = sayi;

    while (normalize > 1)

    {

        kuvvet++;

        normalize /= 10;

    }

    printf("sayının  normalize  hali: %lfx10^%d\n", normalize, kuvvet);

    sayi = kesme(normalize, basamaksistemi);

    printf("sayının   kesilmiş  hali: %gx10^%d\n", sayi, kuvvet);

    sayi = yuvarla(normalize, basamaksistemi);

    printf("sayının yuvarlanmış hali: %gx10^%d\n", sayi, kuvvet);

    return 0;

}

**CALISMA1 – 18.02.2020:**

Verilen 10’luk tabandaki sayiyi 2’lik tabana ceviren C kodunu yaziniz.

#include <stdio.h>

#include <math.h>

/\*

1-) sayı sıfırda while

2-) sayının ikiye bölümünde  kalanı diziye ekle

3-) sayıyı ikiye böl

4-) adım bire git

5-) diziyi ters çevir

\*/

short binaryreverse[32];

short binary[32];

int main(int argc, char const \*argv[])

{

    int decimal = 5;

    int i = 0;

    //döngüde ikilik sayı ters olarak bulunur

    while (decimal != 0)

    {

        binaryreverse[i] = decimal % 2;

        decimal /= 2;

        i++;

    }

    // ters sayımızı düzeltiğimiz kod bloğu

    for (int i = 0; i < 32; i++)

    {

        binary[i] = binaryreverse[31 - i];

    }

    for (int i = 0; i < 32; i++)

    {

        printf("%d", binary[i]);

    }

    return 0;

}

1. **GERCEK SAYILARIN BELLEKTE TUTULMASI**

**ODEV 2 - 25.02.2020:**

64 bitlik sayiyi 10’luk sayiyi cevirme.

#include <stdio.h>

#include <math.h>

char sayi[64];

int main()

{

    //64 bit sayi

    int s;        //işaret biti

    int c = 0;    //üstel bit

    double f = 0; //kesir kısım

    int sayac1 = 0;

    int sayac2 = 0;

    int s1 = 11;

    int s2 = 12;

    printf("çekilecek 64 bit sayiyi giriniz ");

    scanf("%s", &sayi);

    if (sayi[0] == '0')

        s = 0;

    else if (sayi[0 == '1'])

        s = 1;

    while (sayac1 < 11)

    {

        if (sayi[s1] == '1')

            c += pow(2, sayac1);

        s1--;

        sayac1++;

    }

    while (sayac2 < 52)

    {

        if (sayi[s2] == '1')

            f += pow(2, -(sayac2 + 1)); //değer taraf üsü 1 den başlatan sayac2

        s2++;

        sayac2++;

    }

    double sonuc = pow(-1, s) \* pow(2, (c - 1023)) \* (1 + f);

    printf("ondalık sayi=> %f", sonuc);

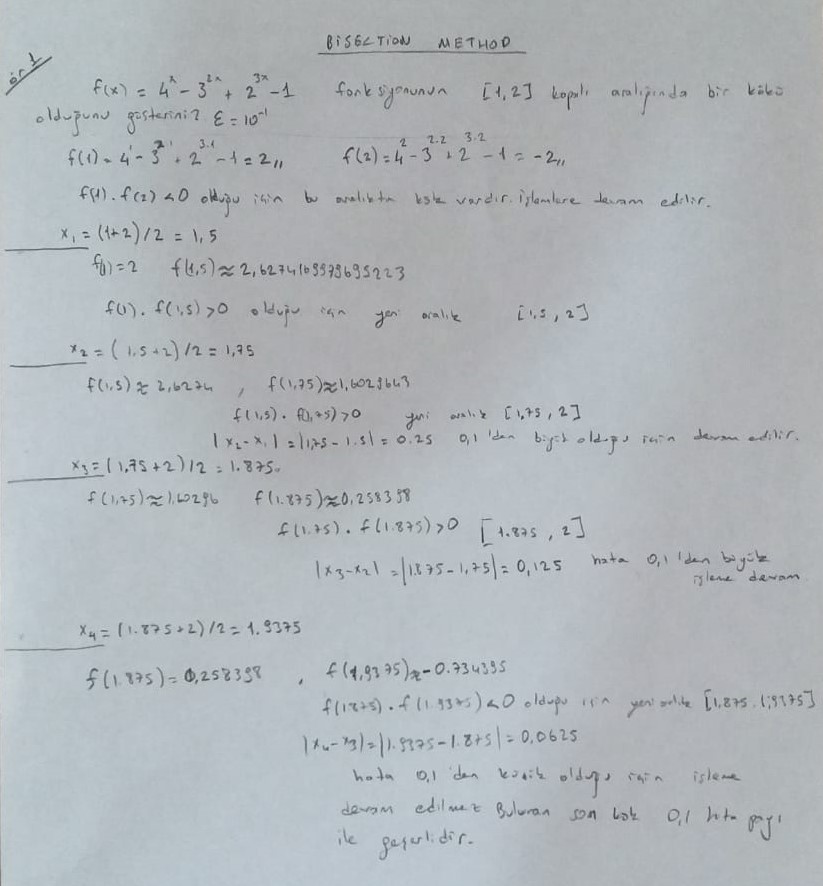
    return 0;

}

**CALISMA2 – 25.02.2020:**

Pow kullanmadan + sonsuz yap. (Double Prec. Format)

1. **BISECTION METOD**



#include <stdio.h>

#include <math.h>

double f(double x)

{

    return pow(4, x) - pow(3, 2 \* x) + pow(2, 3 \* x) - 1;

}

int main(int argc, char const \*argv[])

{

    int iterasyon = 0;

    double xlower = 1.0, xupper = 2.0, xr = 0.0, hata = 0.0, tolerans = 0.1;

    if ((f(xlower) \* f(xupper)) < 0)

    {

        hata = fabs(xupper - xlower);

        double sonkok = xupper;

        while (hata > tolerans)

        {

            iterasyon++;

            xr = (xlower + xupper) / 2;

            if (f(xlower) \* f(xr) < 0)

            {

                xupper = xr;

            }

            else if (f(xlower) \* f(xr) > 0)

            {

                xlower = xr;

            }

            else

            {

                sonkok = xr;

                break;

            }

            hata = fabs(sonkok - xr);

            sonkok = xr;

        }

        printf("Kök %g dir.%d iterasyonda bulundu.", sonkok, iterasyon);

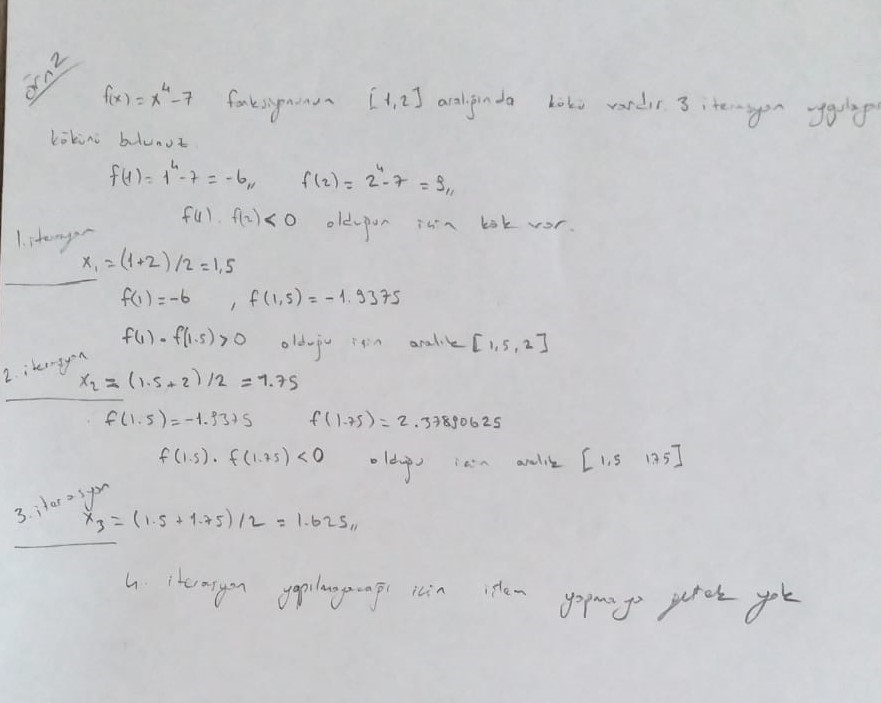
    }

    else

        printf("[%g %g] aralığında kök yoktur", xlower, xupper);

    return 0;

}



#include <stdio.h>

#include <math.h>

double f(double x);

int iterasyononeri(double hatapayi, double aralik1, double aralik2);

int main(int argc, char const \*argv[])

{

    int secim;

    int iterasyon;

    double xlower = 1.0, xupper = 2.0, xr = 0.0, hata = 0.0, tolerans = 0.1;

    printf("\n1.) Kaç iterasyon işlem yapmak istiyorsunuz?\n2.)Hata payı girerek minimum kaç iterasyon yapılmalı öğren\n");

    scanf("%d", &secim);

    switch (secim)

    {

    case 1:

    {

    iterasyongir:

        printf("\nİterasyon sayısı:");

        scanf("%d", &iterasyon);

        goto islem;

    }

    case 2:

    {

        printf("Hata payı: ");

        scanf("%lf", &tolerans);

        printf("\n%lf hata payı için en az %d iterasyon işlem yapılmalıdır", tolerans, iterasyononeri(tolerans, xlower, xupper));

        goto iterasyongir;

        printf("");

    }

    default:

    {

        printf("\nHatalı seçim yatınız.Promram sonlandırıldı.");

        return 0;

    }

    }

islem:

    if ((f(xlower) \* f(xupper)) < 0)

    {

        hata = fabs(xupper - xlower);

        double sonkok = xupper;

        while (iterasyon > 0)

        {

            iterasyon--;

            xr = (xlower + xupper) / 2;

            if (f(xlower) \* f(xr) < 0)

            {

                xupper = xr;

            }

            else if (f(xlower) \* f(xr) > 0)

            {

                xlower = xr;

            }

            else

            {

                sonkok = xr;

                break;

            }

            hata = fabs(sonkok - xr);

            sonkok = xr;

        }

        printf("Kök %g dir.", sonkok);

    }

    else

        printf("[%g %g] aralığında kök yoktur", xlower, xupper);

    return 0;

}

double f(double x)

{

    return pow(x, 4) - 7;

}

int iterasyononeri(double hatapayi, double aralik1, double aralik2)

{

    int n = (log10(fabs(aralik1 - aralik2)) - log10(2 \* hatapayi)) / log10(2);

    /\* +2 olma sebebi

    +1 formüldeki n bulunan değerden büyük olduğu için

    diğer +1 ise formülde hep 1 eksik çıkıyor hatalı yani yada

    formül bisection algoritmasındaki ilk bulduğuz kökü iterasyondan saymıyor.

    (algoritmada ilk kök fonksiyonların çarpımının 0'dan küçük olup olamadığına bakılmadan hesaplanıyor).

    \*/

    return n + 2;

}

1. **TAYLOR ve MACLAURIN SERILERI**

**ODEV 4 – 4. HAFTA:**

f(x) = sinx fonksiyonunu x=0.2 degeri icin maclaurin seri acilimini kullanarak bulunuz. [4 ADIM]

**CALISMA – 4**

f(x)=cosx fonksiyonunu x0=0.23 noktasi civarinda taylor serisini kullanarak 10 adim aciniz. X=0.2 degerini bu acilimi kullanarak bulunuz.

#include <stdio.h>

#include <math.h>

double fak(int a)

{

    if (a == 1 || a == 0)

        return 1;

    else

        return a \* fak(a - 1);

}

int main(int argc, char const \*argv[])

{

    double x = 0.25, toplam = 0.0, turev, x0 = 0.23;

    int k = 0;

    while (k < 10)

    {

        int mod = k % 4;

        if (mod == 0)

        {

            turev = cos(x0);

        }

        else if (mod == 1)

        {

            turev = -sin(x0);

        }

        else if (mod == 2)

        {

            turev = -cos(x0);

        }

        else if (mod == 3)

        {

            turev = sin(x0);

        }

        toplam += (pow(x - x0, k) \* turev) / fak(k);

        k++;

    }

    printf("toplam = %lf ", toplam);

    return 0;

}

1. **NEWTON – RAPHSON METODU**

**ODEV 5 - 5. HAFTA?**

X0 , hata orani ve iterasyon sayisini kullanicidan aldiginiz bir fonksiyonun N-R metodu ile kokune yaklasiniz.

**CALISMA 5:**

f(x)=4\*e-0.5-x seklinde verilen esitligin x0=2 civarindaki kokunu N-R myontemiyle bulunuz.

#include <stdio.h>

#include <math.h>

double f(double x){

    return pow(x, 3)-6\*x+1;

}

double f1(double x){

    return 3\*pow(x, 2)-6;

}

void NewtonRaphson(void){

    double x0=2.0, x=0.0, y=0.0, hata=0.0, tolerans=0.01;

    x=x0-f(x0)/f1(x0);

    hata=fabs(x-x0);

    x0=x;

    printf("Yaklasik Kok -> %lf Yaklasik Hata -> %lf\n\n", x, hata);

    while (hata>tolerans){

        x=x0-f(x0)/f1(x0);

        hata=fabs(x-x0);

        x0=x;

        printf("Yaklasik Kok -> %lf Yaklasik Hata -> %lf\n\n", x, hata);

    }

    printf("Kok -> %lf\n\n", x);

}

int main(){

    NewtonRaphson();

    return 0;

}

1. **KIRIS YONTEMI/SECANT METHOD (6. HAFTA)**

**CALISMA 6**

f(x)=lnx-ex=0 fonksiyonunun 1 araligindaki kokune x0=0.5, x1=0.6 olarak hata toleransi icin secant metoduyla yaklasiniz. 5 ondalik islem yapiniz.

#include <stdio.h>

#include <math.h>

double f(double x)

{

    return log(x) - cos(x);

}

void Secant(void)

{

    double x0 = 1.0, x1 = 1.2, x2 = 0.0, hata = 0.0, tolerans = 0.001;

    x2 = (x0 \* f(x1) - x1 \* f(x0)) / (f(x1) - f(x0));

    hata = fabs(x2 - x1);

    x0 = x1;

    x1 = x2;

    printf("kök-> %1f hata -> %1f \n", x2, hata);

    while (hata > tolerans)

    {

        x2 = (x0 \* f(x1) - x1 \* f(x0)) / (f(x1) - f(x0));

        hata = fabs(x2 - x1);

        x0 = x1;

        x1 = x2;

        printf("kök->%1f hata->%1f \n", x2, hata);

    }

    printf("aranan kök -> %1f hata ->%1f \n", x2, hata);

}

int main()

{

    Secant();

    return 0;

}

1. **DOGRUSAL ENTERPOLASYON (REGULA FALSI METHOD)**

**CALISMA 7**

f(x)=x3\*4x+1=0 denleminin Regula Falsi metodunu kullanarak [0, 1] araliginda kokune yaklasiniz.

#include <math.h>

#include <stdio.h>

#include <conio.h>

#define hata 0.001

#define f(x) x \*x \*x - 4 \* x + 1

int main()

{

    int i = 0;

    float x0, x1, x2, f0, f1, f2;

    printf("\n x0 ve x1 değerlerini girin");

    scanf("%f %f", &x0, &x1);

    do

    {

        f0 = f(x0);

        f1 = f(x1);

        x2 = ((x0 \* f1) - (x1 \* f0) / (f1 - f0));

        f2 = f(x2);

        if (f0 \* f2 < 0)

        {

            x1 = x2;

            f1 = f2;

        }

        else

        {

            x0 = x2;

            f0 \*f2;

        }

        i++;

        printf("\n iterasyon sayısı : %d \t", i);

        printf("\n yaklasık kök %f \t", x2);

        printf("\n fonksiyon değeri : %f \n", f2);

    } while (fabs(f2) > hata);

    getch();

    return 0;

}