**1.** **C++ program for implementation of Bisection Method.**

**Code:**

#include<bits/stdc++.h>

using namespace std;

#define EPSILON 0.001

double func(double x)

{

return x\*x\*x - x - 11;

}

void bisection(double a, double b)

{

if (func(a) \* func(b) >= 0)

{

cout << "You have not assumed right a and b\n";

return;

}

double c = a;

while ((b-a) >= EPSILON)

{

c = (a+b)/2;

if (func(c) == 0.0)

break;

else if (func(c)\*func(a) < 0)

b = c;

else

a = c;

}

cout << "The value of root is : " << c;

}

int main()

{

double a =2, b = 3;

bisection(a, b);

return 0;

}

**Output:**

****

**2. C++ program for implementation of Newton Raphson Method.**

**Code:**

#include<bits/stdc++.h>

#define EPSILON 0.0001

using namespace std;

double func(double x)

{

return x\*x\*x - x - 11;

}

double derivFunc(double x)

{

return 3\*x\*x ;

}

void newtonRaphson(double x)

{

double h = func(x) / derivFunc(x);

while (abs(h) >= EPSILON)

{

h = func(x)/derivFunc(x);

x = x - h;

}

cout << "The value of the root is : " << x;

}

int main()

{

double x0 = 2;

newtonRaphson(x0);

return 0;

}

**Output:**



**3.** **C++ program for implementation of Regular Falsi Method.**

**Code:**

#include<bits/stdc++.h>

using namespace std;

#define MAX\_ITER 1000000

double func(double x)

{

return x\*x\*x - x - 11;

}

void regulaFalsi(double a, double b)

{

if (func(a) \* func(b) >= 0)

{

cout << "You have not assumed right a and b\n";

return;

}

double c = a;

c = (a\*func(b) - b\*func(a))/ (func(b) - func(a));

cout<<"a b f(a) f(b) c f(c)"<<endl;

cout<<a<<" "<<b<<" "<<func(a)<<" "<<func(b)<<" "<<c<<" "<<func(c)<<endl;

for (int i=0; i < MAX\_ITER; i++)

{

if (fabs(func(c))<0.001)

break;

else if (func(c)\*func(a) < 0)

b = c;

else

a = c;

c = (a\*func(b) - b\*func(a))/ (func(b) - func(a));

cout<<a<<" "<<b<<" "<<func(a)<<" "<<func(b)<<" "<<c<<" "<<func(c)<<endl;

}

cout << "The value of root is : " << c;

}

int main()

{

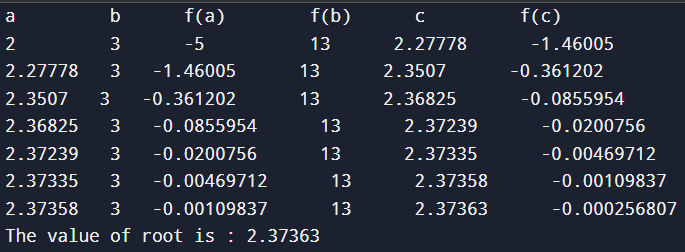
double a =2, b = 3;

regulaFalsi(a, b);

return 0;

}

**Output:**

****

**4. C++ program for implementation of secant method.**

**Code:**

#include<bits/stdc++.h>

using namespace std;

float f(float x)

{

float f = pow(x, 3) - x - 11;

return f;

}

void secant(float x1, float x2, float E)

{

float n = 0, xm, x0, c;

if (f(x1) \* f(x2) < 0) {

do {

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

c = f(x1) \* f(x0);

x1 = x2;

x2 = x0;

n;

if (c == 0)

break;

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

} while (fabs(xm - x0) >=

cout << "Root of the given equation=" << x0 << endl;

cout << "No. of iterations = " << n << endl;

} else

cout << "Can not find a root in the given interval";

}

int main()

{

float x1 = 2, x2 = 3, E = 0.0001;

secant(x1, x2, E);

return 0;

}

**Output:**



**5. C program for implementation of Gauss Jordan method.**

**Code:**

#include<stdio.h>

int main()

{

int i,j,k,n;

float A[20][20],c,x[10];

printf("\nEnter the size of matrix: ");

scanf("%d",&n);

printf("\nEnter the elements of augmented matrix row-wise:\n");

for(i=1; i<=n; i++)

{

for(j=1; j<=(n+1); j++)

{

printf(" A[%d][%d]:", i,j);

scanf("%f",&A[i][j]);

}

}

for(j=1; j<=n; j++)

{

for(i=1; i<=n; i++)

{

if(i!=j)

{

c=A[i][j]/A[j][j];

for(k=1; k<=n+1; k++)

{

A[i][k]=A[i][k]-c\*A[j][k];

}

}

}

}

printf("\nThe solution is:\n");

for(i=1; i<=n; i++)

{

x[i]=A[i][n+1]/A[i][i];

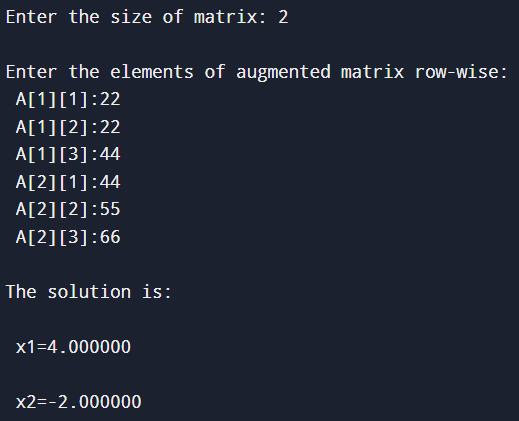
printf("\n x%d=%f\n",i,x[i]);

}

return(0);

}

**Output:**



**6. C program for implementation of Gauss Elimination method.**

**Code:**

#include<stdio.h>

int main()

{

int i,j,k,n;

float A[20][20],c,x[10],sum=0.0;

printf("\nEnter the order of matrix: ");

scanf("%d",&n);

printf("\nEnter the elements of augmented matrix row-wise:\n\n");

for(i=1; i<=n; i++)

{

for(j=1; j<=(n+1); j++)

{

printf("A[%d][%d] : ", i,j);

scanf("%f",&A[i][j]);

}

}

for(j=1; j<=n; j++)

{

for(i=1; i<=n; i++)

{

if(i>j)

{

c=A[i][j]/A[j][j];

for(k=1; k<=n+1; k++)

{

A[i][k]=A[i][k]-c\*A[j][k];

}

}

}

}

x[n]=A[n][n+1]/A[n][n];

for(i=n-1; i>=1; i--)

{

sum=0;

for(j=i+1; j<=n; j++)

{

sum=sum+A[i][j]\*x[j];

}

x[i]=(A[i][n+1]-sum)/A[i][i];

}

printf("\nThe solution is: \n");

for(i=1; i<=n; i++)

{

printf("\nx%d=%f\t",i,x[i]);

}

return(0);

}

**Output:**

