**The Newton Raphson Method**

**Theory**

This method is generally used to obtain the improved result than the Bisection, False position or Iteration method. The method starts with a function *f* defined over the real numbers *x*, the function's derivative *f ′*, and an initial guess *x*0 for a root of the function *f*. If the function satisfies the assumptions made in the derivation of the formula and the initial guess is close, then a better approximation *x*n is

F(Xn-1)

Xn = Xn-1 - -------------

F`(Xn-1)

The process is repeated as until a sufficiently accurate value is reached.

**Code**

#include<bits/stdc++.h>

using namespace std;

#define error 0.000001

double f(double x)

{

return (3\*x-1-cos(x));

}

double fdiff(double x)

{

return (3+sin(x));

}

void NewtonRaphson(double x)

{

int i=0;

double m,xn;

do

{

m = x;

xn = x - ( (f(x))/fdiff(x) );

x = xn;

cout<<" x"<<i<<" : "<<xn<<" Error Rate: "<<fabs(xn-m)<<endl;

i++;

}while(fabs(xn-m)>=error);

cout<<"The actual value of x is: "<<xn<<endl;

}

int main()

{

double x;

cout<<"Enter the value of x: ";

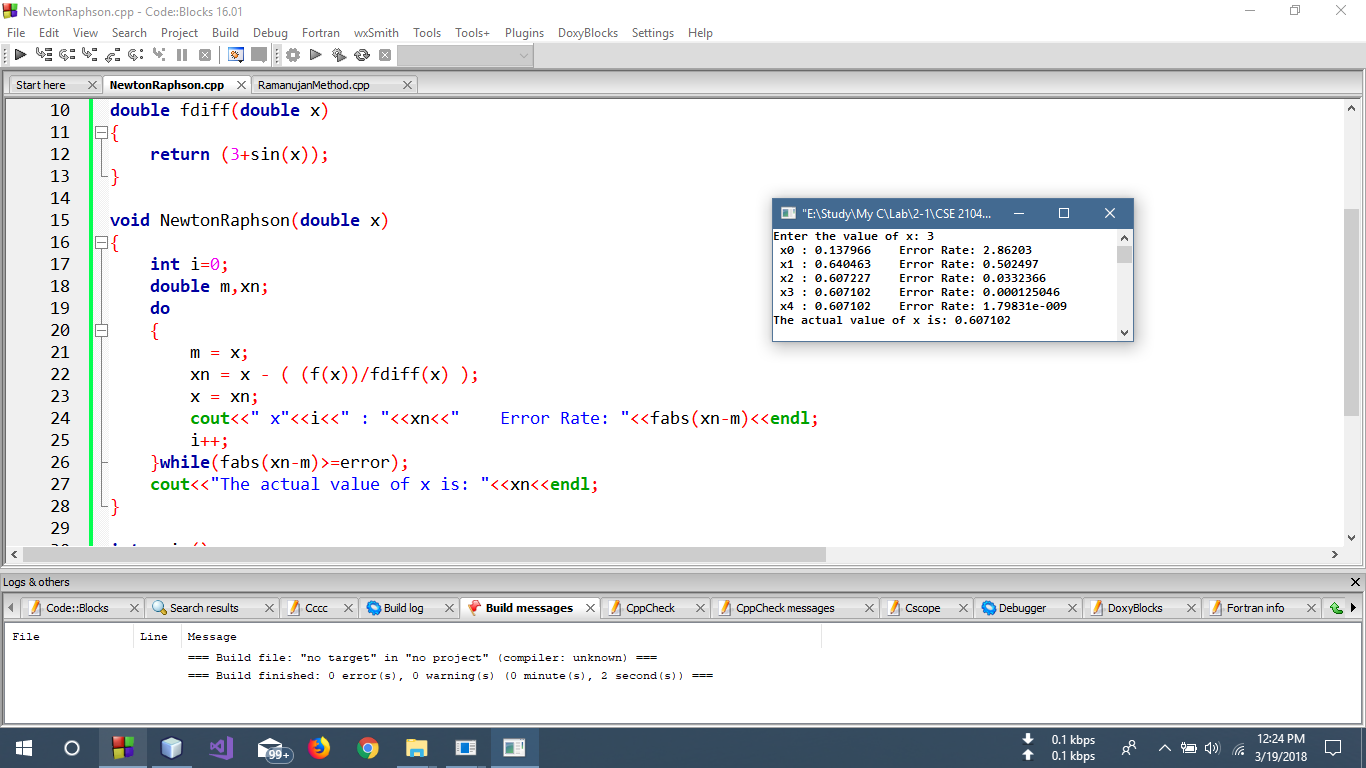
cin>>x;

NewtonRaphson(x);

return 0;

}

**Output**



**Discussion**

Here in the above code, two functions named f() and fdiff() were declared that returned double type value of the function for the parameter x. Then in the main() function, in a do while loop, the general formula of Newton-Raphson method was applied until the value of root x had the error rate of 0.000001.

**The Ramanujan Method**

**Theory**

Srinivasa Ramanujan (1887 – 1920) described an iterative method which can be used to determine the smallest root of the equation f(x) = 0 where f(x) is the form of,

f(x) = 1 – (a1x + a2x^2 + a3x^3 + a4x^4 + ……) ………(1)

for smaller values of x , it can be written as,

(1 – (a1x + a2x^2 + a3x^3 + a4x^4 + ……))^(-1) = ( b1 + b2x + b3x^2 + …… ) …..(2)

Expanding the left side of the equation (2) TO binomial theorem, it can be obtained that,

1 + (a1x + a2x^2 + a3x^3 + …..) + (a1x + a2x^2 + a3x^3 + ….)^2 + … = b1 + b2x + b3x^2 +...

…(3).

Comparing the co-efficients of like powers of x on both sides of equation no. (3), it is obtained that,

b1 = 1,

b2 = a1 = a1b1,

b3 = a1b2 + a2b1,

b4 = a1b3 + a2b2 + a3b1,

.

.

bn = a1bn-1 + a2bn-2 + ….. + an-1b1 …..(4) ; n = 2,3,4…..

Without any proof Ramanujan states that, the successive convergents, (bn/bn+1) is the root of equation f(x) = 0.

**Code**

#include<bits/stdc++.h>

using namespace std;

void Ramanujan(double a1,double a2,double a3)

{

double a[100],b[100];

a[0]=1.0;

a[1]=a1;

a[2]=a2;

a[3]=a3;

for(int i=4;i<38;i++)

a[i]=0;

b[0]=1.0;

b[1]=1.0;

int i,j,k,x=1,y=2,m,count=0;

i=1,j=1;

int n=2;

double result,temp,error,sum=0;

cout<<"b"<<"1"<<" = "<<b[1]<<endl;

while(n<18)

{

count++;

i=1,j=1;

while(1)

{

sum += a[j]\*b[n-i];

if((n-i)==1 && j==(n-1)){

break;

}

i++;

j++;

}

if((n-i)==1 && j==(n-1)){

b[n]=sum;

cout<<"b"<<n<<" = "<<b[n]<<endl;

n++;

sum=0;

}

result = (double)(b[x]/b[y]);

if(count==1)

temp=result;

else

error = abs(result-temp);

cout<<"b"<<x<<"/"<<"b"<<y<<" = "<<result<<" Error: "<<error<<endl;

temp = result;

m=y;

x=y;

y=m+1;

if(error==0.00001)

break;

}

cout<<"\nThe Smallest Root is: "<<result<<endl;

}

int main()

{

double a1,a2,a3;

cout<<"Enter the value of a1: ";

cin>>a1;

cout<<"Enter the value of a2: ";

cin>>a2;

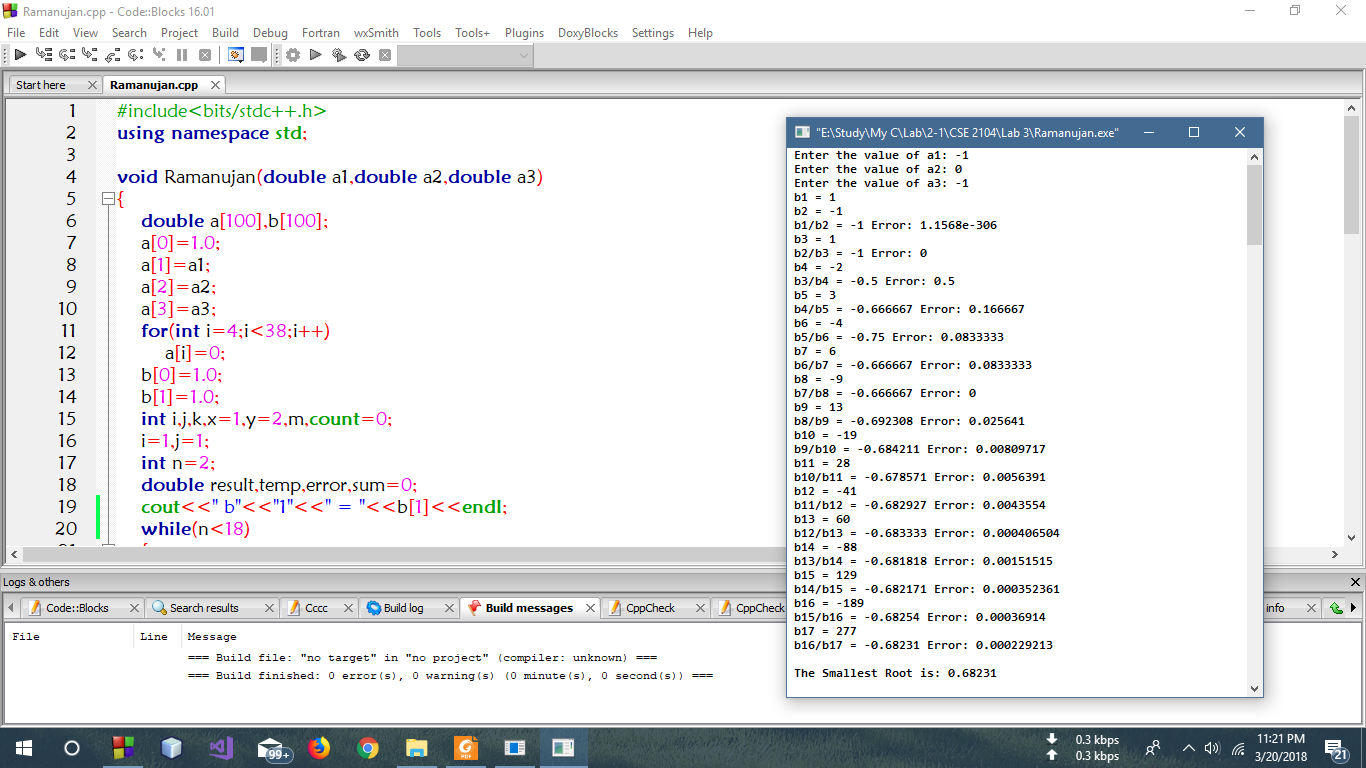
cout<<"Enter the value of a3: ";

cin>>a3;

Ramanujan(a1,a2,a3);

return 0;

}

**Output**

**Discussion**

Here in the above code, a function named Ramanujan() was declared where a1,a2,a3 went as parameters denoting the co-efficient of the x,x^2,x^3 respectively. In that very function, the equation no (4) from theory section was performed using loop and conditional logic. Thus the error rate was shown in output along with the roots. And finally the smallest root was acquired and shown in the output.