**ASSIGNMENT - 1**

**Objective : Write a Program to Implement Bisection Method.**

Initialize itr

a=x

b=x

Call function bisect With x1,a,b,itr

Print itr,x1

Call function bisectWith x,a,b,itr

Define function f(x)

Define function bisect

B

B

Get the values of a,b,aerr,maxitr

Print ‘Solution does not converge

Print itr,x1

Itr=itr+1

X=(a+b)/2.0

Is itr<maxitr

Is fabs(x1-x)<aerr ?

Is f(a)\*f(x)<0 0000

10

20

20

10

B

**Flow Chart :**

**Algorithm:**

**STEP 1 :** Start

**STEP 2 :** Defining functions f(x) & bisect

**STEP 3 :** Read: a, b, aerr, maxitr

**STEP 4 :** Initializing itr

**STEP 5 :** Calling function bisect

**STEP 6 :** Set x=(a+b)/2.0 and itr=itr +1

**STEP 7 :**  Print: itr,x1

**STEP 8 :** Check f(a)\*f(x)<0

If

(a) yes: Set b=x

(b) No:Set a=x

**STEP 9 :** Repeat step 5 to 7

**STEP 10 :** Check fabs(x1-x)<aer

If

(a) yes: print itr,x1 and go to step 11

(b) No: check itr<maxitr

(i) yes: repeat step 8 to 10

(ii)No: print: ‘Solution does not converge’ and go to step 11

**STEP 11 :** Exit.

**C++ Program:**

#include<iostream>

#include<iomanip>

#include<math.h>

using namespace std;

float f(float x)

{

return (x\*x\*x - 4\*x - 9); }

void bisect(float \*x,float a,float b,int \*itr)

{

\*x=(a+b)/2;

++(\*itr);

cout<<"Iteration no." <<setw(3) << \*itr <<"X= "<< setw(7) << setprecision(5)<<\*x <<endl ;

}

int main()

{

int itr = 0, maxitr;

float x, a, b, aerr, x1;

cout<<"Name: Sahil Sao"<<"\nclass:MCA II"<<"\nPath:D:/ Sahil / bisection.cpp"<<endl;

cout<< "Enter the value of a,b,"<< "allowed error, maximum iterations" << endl ;

cin>>a>>b>>aerr>>maxitr;

cout<<"fixed";

bisect(&x,a,b,&itr);

do

{

if (f(a)\*f(x) <0)

b=x;

else

a=x;

bisect (&x1,a,b,&itr);

if(fabs(x1-x) < aerr)

{

cout<<"After" << itr << "iterations,root"<<"=" << setw(6) << setprecision(4) <<x1;

return 0;

}

x=x1;

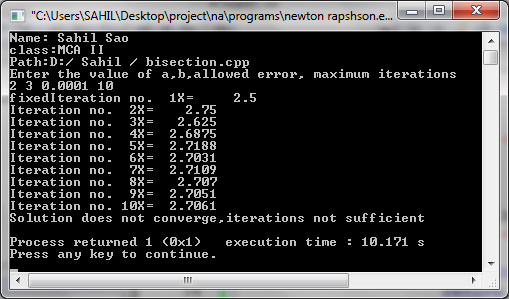
}while (itr <maxitr);

cout<< "Solution does not converge,"<< "iterations not sufficient" <<endl;

return 1

}

**Output:**



**ASSIGNMENT- 2**

**Objective : Write a Program to Implement Method Newton's Raphson .**

Loop for itr=1 to maxitr

Define function df(x)

Get the values of

x0,aerr,maxitr

Print Solution

Is fabs(h)<aerr

x0=x1

h=f(x0)/df(x0) x1=x0-h

Print itr,x1

Define function f(x)

**Flow Chart:**

**Algorithm:**

**STEP 1:** Start

**STEP 2:** Defining functions f(x) & df(x)

**STEP 3:** Read: x0,aerr,maxitr

**STEP 4:** Repeat step 5 to 7 for itr=1 to maxitr

**STEP 5:** Set h=f(x0)/df(x0)

x1=x0-h

**STEP 6:** Print: itr and x1

**STEP 7:** Check fabs(h)<aer

If

(a) yes: print solution and go to step 8

(b) No: set x0=x1,print: ‘Solution does not converge’ and go to Step 8

**STEP 8:** EXIT.

**C++ Program:**

#include<iostream>

#include<iomanip>

#include<math.h>

using namespace std;

float f(float x)

{

return x\*log10(x)-1.2;

}

float df(float x)

{

return log10(x)+0.43429;

}

int main()

{

int itr,maxitr;

float h,x0,x1,aerr;

cout<<"Name: Sahil Sao"<<"\nclass: MCAII"<<"\nPath: D:/Sahil/NR.cpp"<<endl;

cout<< "Enter x0,allowed error,"<< "maximum iterations" << endl;

cin >> x0 >> aerr >> maxitr;

cout <<" fixed ";

for (itr=1;itr<=maxitr;itr++)

{

h=f(x0)/df(x0);

x1=x0-h;

cout << "Iteration no." << setw(3) << itr<< "X= "<< setw(9)<<setprecision(6)<<x1<< endl;

if (fabs(h) < aerr)

{

cout << "after no." << setw(3) << itr<< "iterations, root = "

<< setw(8) << setprecision(6) << x1;

return 0;

}

x0=x1;

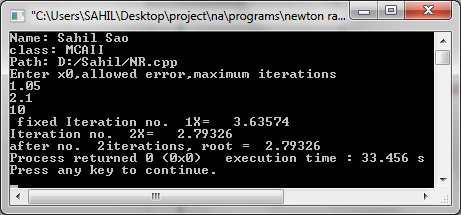
}

cout << "Iteratiions not sufficient,"<< "solution does not converge" << endl;

return 1;

}

**Output:**



**ASSIGNMENT-3**

**Objective: Write a Program to Implement Regula Falsi Method.**

Initialize itr

x0=x2

x1=x22

Call function Regulawith x3,x0,x1,f(x0),f(x1),itr

Call function regulawith x2,x0,x1,f(x0),f(x1),itr

Define function f(x)

Define function regula

R

Get the values of x0,x1,aerr,maxitr

Print Solution does not converge

Print Solution

x=x0-((x1-x0)/(fx1-fx0))\*fx0

Is itr<maxitr

Is fabs(x3-x2)<aerr ?

Is f(x0)\*f(x2) <0

10

20

20

R

x2=x3

20

Print itr,x

**Flow Chart:**

**Algorithm:**

**STEP 1 :** Start

**STEP 2 :** Defining functions f(x) & regula

**STEP 3 :** Read: x0,x1,aerr,maxitr

**STEP 4 :** Initializing itr

**STEP 5 :** Calling function regula

**STEP 6 :** Set x=x0-(x1-x0)/f(x1)-f(x0)\*f(x0)

**STEP 7 :** Print: itr,x

**STEP 8 :** Check f(x0)\*f(x2)<0

If

(a) yes: Set x1=x2

(b) No:Set x0=x2

**STEP 9 :** Repeat step 5 to 7

**STEP 10 :** Check fabs(x3-x2)<aerr

If

(a) yes: print itr,x1 and go to step 11

(b) No: check itr<maxit

(i)yes: repeat step 8 to 10

(ii)No: print: ‘Solution does not converge’ and go to step 11

**STEP 11 :** Exit

**C++ Program:**

#include<iostream>

#include<iomanip>

#include<math.h>

using namespace std;

float f(float x)

{

return cos(x)-x\*exp(x);

}

void regula (float \*x,float x0, float x1,float fx0,float fx1,int \*itr )

{

\*x=x0-((x1-x0)/(fx1-fx0))\*fx0;

++(\*itr);

cout<<"Iteration no." <<setw(3) << \*itr<<"X= " << setw(7) << setprecision(5)<<\*x << endl;

}

int main()

{

int itr=0, maxitr;

float x0,x1,x2,x3,aerr;

cout<<"Name: Sahil Sao"<<"\nclass: MCAII"<<"\nPath: D:/Sahil/RF.cpp";

cout<< "\nEnter the values for x0,x1,"<<"allowed error,maximum iterations"<<endl;

cin >> x0 >> x1 >> aerr >> maxitr;

regula(&x2,x0,x1,f(x0),f(x1),&itr);

cout <<" fixed ";

do

{

if (f(x0)\*f(x2) < 0)

x1=x2;

else

x0=x2;

regula(&x3,x0,x1,f(x0),f(x1),&itr);

if (fabs(x3-x2) < aerr)

{

cout<<"After"<<itr<<"iterations,"<<"root = " << setw(6) <<setprecision(4)<< x3 << endl; return 0;

}

x2=x3;

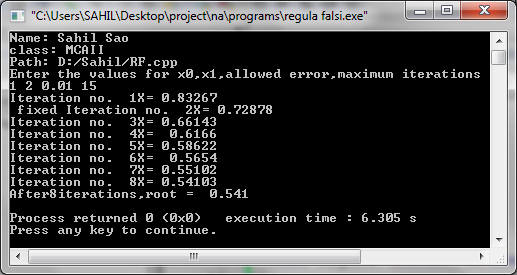
} while(itr < maxitr);

cout << "Solution does not converge,"<< "iterations not sufficient" << endl;

return 1;

}

**Output:**



**ASSIGNMENT-4**

**Objective: Write a Program to Implement Gauss sidle interactive method.**

**Flow chart:**

Get the Augmented Matrix in Array A

x0,aerr,maxitr

Print Solution

End Loop (j)

End Loop (i)

End Loop (k)

a[i][k]-=a[j][k]\*t

Loop for k=0 to N

t=a[i][j]/a[j][j]

Is i!=j ?

Loop for i=0 to N-1

Loop for j=0 to N-1

**Algorithm:**

**STEP 1:** Start

**STEP 2:** Read the element of matrix

**STEP 3:** Repeat step 4 and 5 for j=0 to N-1

**STEP 4:** Repeat step 5 for i=0 to N-1

**STEP 5:** Check i ≠ j

If

(a) yes: (i) set t=a[i][j]/a[j][j]

(ii) Repeat

(iii) for k=0 to N

(iv) a[i][k]=a[i][k]-a[j][k]\*t

(b) No: go to Step 6

**STEP 6:** Print solution

**STEP 7:** Exit.

**C++ Program:**

#include<iostream>

#include<conio.h>

#include<iomanip>

#include<math.h>

#define N 3

using namespace std;

int main()

{

float a[N][N+1],x[N],aerr,maxerr,t,s,err,fixed;

int i,j,itr,maxitr;

for(i=0;i<N;i++)

x[i]=0;

cout<<"\nName:Sahil Sao"<<"\nclass:Mca II"<<"\nPath:D:/Sahil/GousSei.cpp";

cout<<"\nEnter the Element of the"<<" Argumant Matrix Rowwise"<<endl;

for(i=0;i<N;i++)

for(j=0;j<N+1;j++)

cin>>a[i][j];

cout<<"Enter the allow error"<<"Maximum iteration"<<endl;

cin>>aerr>>maxitr;

cout<<"Iteration"<<setw(6)<<"x[1]"<<setw(11)<<"x[2]"<<setw(11)<<"x[3]"<<endl;

for(itr=1;itr<=maxitr;itr++)

{

maxerr=0;

for(i=0;i<N;i++)

{

s=0;

for(j=0;j<N;j++)

if(j!=i)

s+=a[i][j]\*x[j];

t=(a[i][N]-s)/a[i][i];

err=fabs(x[i]-t);

if(err>maxerr)

maxerr=err;

x[i]=t;

}

cout<<setw(5)<<itr;

for(i=0;i<N;i++)

cout<<setw(11)<<setprecision(4)<<x[i];

cout<<endl;

if(maxerr<aerr)

{

cout<<"Converges in"<<setw(3)<<itr<<"iteration"<<endl;

for(i=0;i<N;i++)

cout<<"X["<<setw(3)<<i+1<<"]="<<setw(7)<<setprecision(4)<<x[i]<<endl;

return 0;

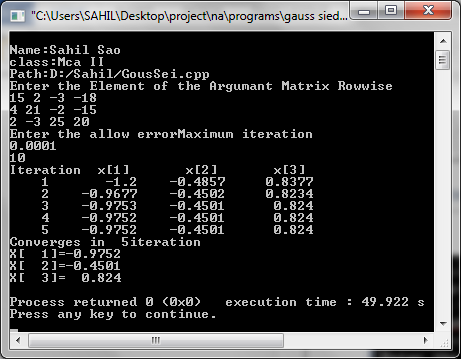
}

}

cout<<"Solution does not converge,"<<"Iteration not sufficient"<<endl; return 1;

}

**Output:**



**ASSIGNMENT- 5**

**Objective: Write a Program to Implement Gauss Elimination Method.**

**Flow Chart:**

Get the Augmented Matrix in Array a

**Algorithm:**

End Loop (k)

X[i]=(a[i][N]-s)/a[i][i]

End Loop(j)

S+=a[i][j]\*x[j]

Loop for j=0 to n-2

Loop for i=j+1 to N-1

t=a[i][j]/a[j][j]

Loop for k=0 to N

a[i][k]-=a[j][k]\*t

End Loop (i)

End Loop (j)

S=0

Loop for i=N-1 to 0 Step-1

Loop for j=i+1 to n-1

**STEP 1 :** Start

**STEP 2 :** Read the element of matrix

**STEP 3 :** Repeat step 4 to 7 for j=0 to n-2

**STEP 4 :** Repeat step 5 to 7 for i=j+1 to n-1

**STEP 5 :** Set t=a[i][j]/a[j][j]

**STEP 6 :** Repeat step 7 for k=0 to N

**STEP 7 :** a[i][k]-=a[j][k]\*t

**STEP 8 :** Repeat step 9 to 2 for i=N-1 to 0

**STEP 9 :** Set: S=0

**STEP 10 :** Repeat step 11 for j=i+1 to n-1

**STEP 11 :** Set: S=S+a[i][j]\*x[j]

**STEP 12 :** Set x[i]=(a[i][N]-s)/a[i][j]

**STEP 13 :** Print solution

**STEP 14 :** Exit.

**C++ Program:**

# include <iostream>

# include <iomanip>

# include <math.h>

#include <conio.h>

# define N 3

using namespace std;

int main()

{

float a[N][N+1],x[N],t,s,fixed;

int i,j,k;

cout<<"Name: Sahil Sao \nclass: MCAII"<<"\nPath: D:/Sahil/GousEli.cpp";

cout<<"\n\n ENTER THE ELEMENTS OF THE AGUMENTED MATRIX ROW-WISE \n\n"<<endl;

for (i=0;i<N;i++)

for (j=0;j<N+1;j++)

cin>>a[i][j];

for(j=0;j<N-1;j++)

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

{

t=a[i][j]/a[j][j];

for(k=0;k<N+1;k++)

a[i][k]-=a[j][k]\*t;

}

cout<<"\n THE UPPER TRINGULER MATRIX IS:= \n\n";

for(i=0;i<N;i++)

{

for(j=0;j<N+1;j++)

cout<<"\t"<<setprecision(4)<<a[i][j]; cout<<"\n";

}

for(i=N-1;i>=0;i--)

{

s=0;

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

s+=a[i][j]\*x[j];

x[i]=(a[i][N]-s)/a[i][i];

}

cout<<"\n THE SOLUTION IS :- "<<endl;

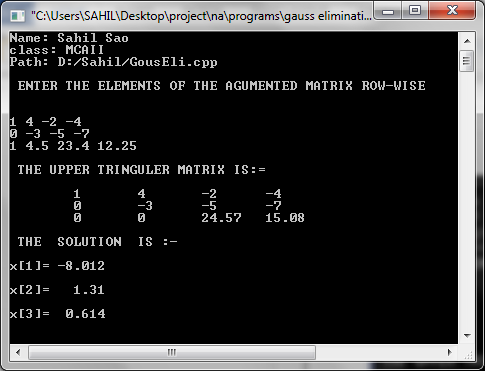
for(i=0;i<N;i++)

cout<<"\nx["<<setw(1)<<i+1<<"]="<<setw(7)<<setprecision(4)<<x[i]<<endl;

getch();

}

**Output:**



**ASSIGNMENT-6**

**Objective: Write a Program to Implement Gauss Jordon method.**

Loop for j=0 to N-1

Get the Augmented Matrix in Array A

x0,aerr,maxitr

Is i!=j ?

t=a[i][j]/a[j][j]

Loop for i=0 to N-1

Loop for k=0 to N

Print Solution

a[i][k]-=a[j][k]\*t

End Loop (k)

End Loop (i)

End Loop (j)

**Flow Chart:**

**Algorithm:**

**STEP 1:** Start

**STEP 2:** Read the element of matrix

**STEP 3:** Repeat step 4 and 5 for j=0 to N-1

**STEP 4:** Repeat step 5 for i=0 to N-1

**STEP 5:** Check i ≠ j

If

(a) yes: (i) set t=a[i][j]/a[j][j]

(ii) Repeat

(iii) for k=0 to N

(iv) a[i][k]=a[i][k]-a[j][k]\*t

(b) No: go to Step 6

**STEP 6:** Print solution

**STEP 7:** Exit.

**C++ Program:**

#include<iostream>

#include<iomanip>

#define N 4

using namespace std;

int main()

{

float a[N][N+1],t;

int i,j,k;

cout<<"Name: Sahil Sao \n Class:MCAII"<<"\nPath:d:/Sahil/Gjor.cpp";

cout<< "/nEnter the elements of the "<<" augmented matrix rowwise " << endl;

for(i=0;i<N;i++)

for(j=0;j<N+1;j++)

cin>> a[i][j];

/\* now calculating the values of x1,x2,.......,xN \*/

cout << "fixed";

for(j=0;j<N;j++)

for(i=0;i<N;i++)

if(i!=j)

{

t=a[i][j]/a[j][j];

for(k=0;k<N+1;k++)

a[i][k] -= a[j][k]\*t;

}

/\* now printing the diagonal matrix \*/

cout<< "The diagonal matrix is :-" <<endl;

for(i=0;i<N;i++)

{

for(j=0;j<N+1;j++)

cout<< setw(9) << setprecision(4) <<a[i][j];

cout<<endl;

}

/\* now printing the results \*/

cout<< "The solution is :- " << endl;

for(i=0;i<N;i++)

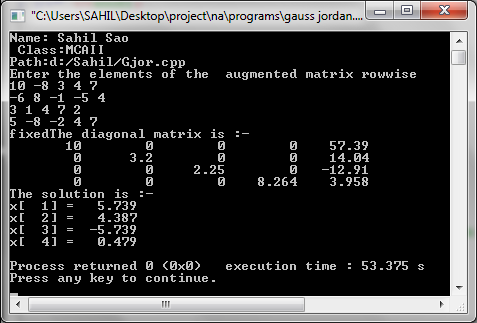
cout<< "x["<<setw(3) << i+1 << "] = "<< setw(7) << setprecision(4)

<< a[i][N]/a[i][i] << endl;

return 0;

}

**Output:**



**ASSIGNMENT-7**

**Objective: Write a Program to implement factorized method.**

**Flow Chart:**

Calculate elements of l & u

Print Array x as Solution

Find v by solving lv=b by forward Substitution

Print elements of l & u

Find x by solving ux=v by backward Substitution

Get the elements of Augmented Matrix into

Arrays a & b

**Algorithm:**

**STEP 1:** Start

**STEP 2:** Read the element of matrix

**STEP 3:** Calculate elements of l & v

**STEP 4:** Print elements

**STEP 5:** Find V by solving lv=b by forward substitution

**STEP 6:** Find X by solving UX=V by backward substitution

**STEP 7:** Print array X

**STEP 8:** Exit.

**C++ Program:**

#include<iostream>

#include<conio.h>

#include<iomanip>

#include<math.h>

#define N 3

using namespace std;

typedef float matrix[N][N];

matrix l,u,a;

float b[N],x[N],v[N];

void urow(int i)

{

float s;

int j,k;

for(j=i;j<N;j++)

{

s=0;

for(k=0;k<N-1;k++)

s+=u[k][j]\*l[i][k];

u[i][j]=a[i][j]-s;

}

}

void lcol(int j)

{

float s;

int i,k;

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

{

s=0;

for(k=0;k<=j-1;k++)

s+=u[k][j]\*l[i][k];

l[i][j]=(a[i][j]-s)/u[j][j];

}

}

void printmat(matrix x)

{

int i,j;

for(i=0;i<N;i++)

{

for(j=0;j<N;j++)

cout<<setw(8)<<setprecision(4)<<x[i][j];

cout<<endl;

}

}

int main()

{

int i,j,m,fixed;

float s;

clrscr();

cout<<"Name: Sahil Sao"<<"\n Class: MCA II"<<"\n Path: d:/Sahil/Factorized.cpp";

cout<<"\n ENTER THE ELEMENTS OF AUGUMENTED MATRIX ROW-WISE\n\n";

for(i=0;i<N;i++)

{

for(j=0;j<N;j++)

cin>>a[i][j];

cin>>b[i];

}

cout<<fixed;

for(i=0;i<N;i++)

l[i][i]=1.0;

for(m=0;m<N;m++)

{

urow(m);

if(m<N-1)

lcol(m);

}

cout<<setw(30)<<"\n\n UPPER MATRIX IS :-"<<endl;

printmat(u);

cout<<setw(25)<<"\n\n LOWER MATRIX IS:-"<<endl;

cout<<"\n";

printmat(l);

for(i=0;i<N;i++)

{

s=0;

for(j=0;j<=i-1;j++)

s+=l[i][j]\*v[j];

v[i]=b[i]-s;

}

for(i=N-1;i>=0;i--)

{

s=0;

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

s+=u[i][j]\*x[j];

x[i]=(v[i]-s)/u[i][i];

}

cout<<"\n\n THE SOLUTION IS :- "<<"\n";

for(i=0;i<N;i++)

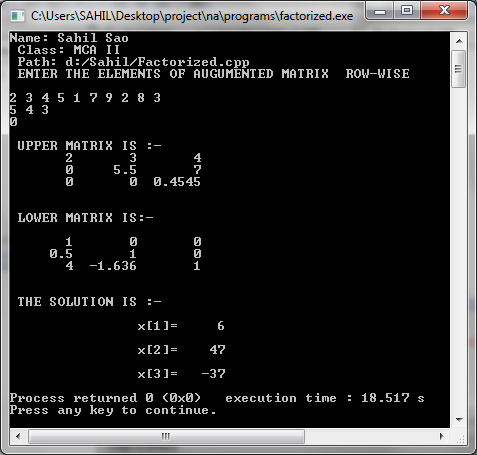
cout<<"\n\t\tx["<<setw(1)<<i+1<<"]="<<setw(6)<<setprecision(4)

<<x[i]<<endl

return 0;

}

**Output:**



**ASSIGNMENT-8**

**Objective: Write a Program to implement power method.**

**Flow Chart:**

e=t

Print results of Iteration

Is (errv<aerr)

& (maxe<aerr)

End Loop (itr)

Print Solution does not converge

Print Solution

**Algorithm:**

**STEP 1 :** Start

**STEP 2 :** Input a, x,aerr,maxitr

**STEP 3 :** Define subroutine findmax

**STEP 4 :** Call findmax(&e,x,n)

(a)max = fabs(x(1))

(b)Loop for i=1 to N-1

(c)If fabs(x[i]>max )then

max= fabs(x[i])

[End loop(i)]

**STEP 5 :** Loop for itr=1 to maxitr

**STEP 6 :** Calculate r=a\*x

**STEP 7 :** Call findmax(&t,r,n)

**STEP 8 :** Repeat step(4)

**STEP 9 :** Normalize r

**STEP 10 :** maxe=0

**STEP 11 :** Loop for i=0 to N-1

**STEP 12 :** err= fabs(x[i]-r[i])

**STEP 13 :** If err>maxe Then

set maxe=err

**STEP 14 :** x[i]=r[i]

[End loop(i)]

**STEP 15 :** err=fabs(t-e)

**STEP 16 :** set e=t

**STEP 17 :** Print result y iteration

**STEP 18 :** If(err<aerr)&(maxe<aerr)

Print solution

[End loop (itr)]

**STEP 19 :** Print solution does not converge

**STEP 20 :** Exit

**C++ Program:**

#include<iostream>

#include<conio.h>

#include<stdio.h>

#include<math.h>

#include<process.h>

#define N 3

using namespace std;

void findmax(float \*max,float x[N])

{

int i;

\*max=fabs(x[0]);

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

{

if(fabs(x[i])>\*max)

\*max=fabs(x[i]);

}

}

void main()

{

float a[N][N],x[N],r[N],aerr,err,s,e,errv,t,maxe;

int maxitr,i;

cout<<"Name: Sahil Sao"<<"\n Class: MCA II"<<"\n Path: d:/Sahil/Power.cpp";

cout<<"\n\nEnter the matrix ";

for(int i=0;i<N;i++)

{

for(int j=0;j<N;j++)

{

cin>>a[i][j];

}

}

cout<<"\n\nEnter the initial approximation :";

for(i=0;i<N;i++)

{

cin>>x[i];

}

cout<<"\n\nEnter allowed error :";cin>>aerr;

cout<<"\n\nEnter Maximum iteration : ";cin>>maxitr;

cout<<"\n\n\nITR no. \tEigen value\t\tEigenVector\n";

findmax(&e,x);

for(int itr=1;itr<=maxitr;itr++)

{

for(i=0;i<N;i++)

{

s=0;

for(int j=0;j<N;j++)

{

s=s+a[i][j]\*x[j];

}

r[i]=s;

}

findmax(&t,r);

for(i=0;i<N;i++)

r[i]=r[i]/t;

maxe=0;

for(i=0;i<N;i++)

{

err=fabs(x[i]-r[i]);

if(err>maxe)

maxe=err;

x[i]=r[i];

}

errv=fabs(t-e);

e=t;

cout<<"itr "<<itr<<"\t\t"<<e;

for(i=0;i<N;i++)

cout<<"\t\t"<<x[i];

cout<<"\n";

if((errv<=aerr)&&(maxe<=aerr))

{

cout<<"\nconverges in "<<itr<<" itration";

cout<<"\nLargest eigen value = "<<e<<"\n";

cout<<"Eigen value :- \n";

for(i=0;i<N;i++)

{

cout<<"X["<<i+1<<"] = "<<"\t\t"<<x[i]<<"\n";

}

cout<<"\n";

getch();

exit(0);

}

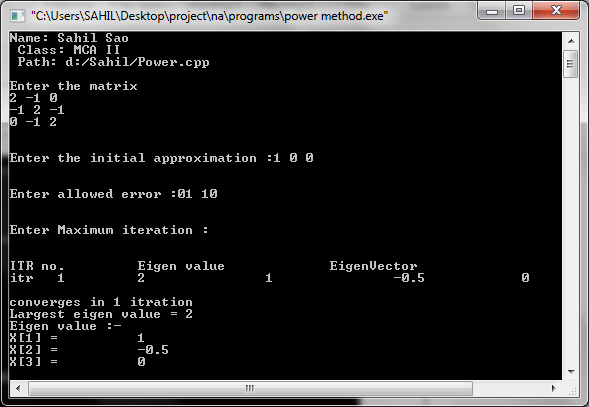
}

cout<<"solution not coverage";

getch();

}

**Output:**



**ASSIGNMENT-9**

**Objective: Write a Program to implement list squares method.**

**Flow Chart:**

Initialize all elements of augm

to zero

Get the value of n

Read in the data points and increment appropriate elements of augm

Assign values to non-unique elements of augm

Print augm

Solve for a,b,c by Gauss Jordan

Method

Print a,b,c as Solution

**Algorithm:**

**STEP 1 :** Start.

**STEP 2 :** Initialize all elements ofaugm to zero.

**STEP 3 :** Input n.

**STEP 4 :** Read in the data points and increment appropriate elements of augm.

**STEP 5 :** Assign values to non-unique elements of augm.

**STEP 6 :** Print augm.

**STEP 7 :** Solve for a,b,c by gauss Jordan method.

**STEP 8 :** Print a,b,c as solution.

**STEP 9 :** Exit.

**C++ Program:**

#include<iostream.h>

#include<iomanip.h>

int main()

{

float augm[3][4]={{0,0,0,0},{0,0,0,0},{0,0,0,0}};

float t,a,b,c,x,y,xsq;

int i,j,k,n;

cout<<"\n Name: Sahil Sao"<<"\n Class: MCA II"<<"\n Path: d:/Sahil/LeastSqu.cpp";

cout<<"\nEnter the no. of pairs of"<<"observed values:"<<endl;

cin>>n;

augm[0][0]=n;

for(i=0;i<n;i++)

{

cout<<"Pair no. "<<i+1<<endl;

cin>>x>>y;

xsq=x\*x;

augm[0][1]+=x;

augm[0][2]+=xsq;

augm[1][2]+=x\*xsq;

augm[2][2]+=xsq\*xsq;

augm[0][3]+=y;

augm[1][3]+=x\*y;

augm[2][3]+=xsq\*y;

}

augm[1][1]=augm[0][2];

augm[2][1]=augm[1][2];

augm[1][0]=augm[0][1];

augm[2][0]=augm[1][1];

cout<<"The augmented matrix is :-"<<endl;

for(i=0;i<3;i++)

{

for(j=0;j<4;j++)

cout<<setw(9)<<setprecision(4)<<augm[i][j];

cout<<endl;

}

/\*Now solving for a,b,c by Gauss Jordan Method\*/

for(j=0;j<3;j++)

for(i=0;i<3;i++)

if(i!=j)

{

t=augm[i][j]/augm[j][j];

for(k=0;k<4;k++)

augm[i][k]-=augm[j][k]\*t;

}

a=augm[0][3]/augm[0][0];

b=augm[1][3]/augm[1][1];

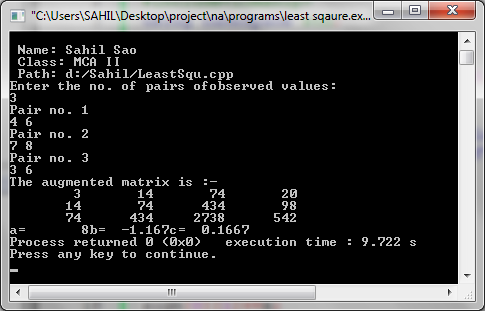
c=augm[2][3]/augm[2][2];

cout<<setprecision(4)<<"a="<<setw(8)<<a<<"b="<<setw(8)<<b<<"c="<<setw(8)<<c;

return 0;

}

**Output:**



**ASSIGNMENT-10**

**Objective: Write a Program to implement Lagrange’s interpolation method.**

**Flow Chart:**

y=0

Print x, y as Solution

nr=dr=1

Loop for i=0 to n

Loop for J=0 to N

Get the value of n

Get elements of ax, ay

Get the value of x

Is j!=i

nr\*=x-ax[j]

dr\*=ax[i]-ax[j]

End Loop (j)

y+=(nr/dr)\*ay[i]

End Loop (i)

**Algorithm:**

**STEP 1 :** Start.

**STEP 2 :** Input n.

**STEP 3 :** Input ax,ay.

**STEP 4 :** Input x.

**STEP 5 :** set y=0.

**STEP 6 :** Loop for i=0 to n.

**STEP 7 :** set nr=dr=1.

**STEP 8 :** Loop for j=0 to n.

**STEP 9 :** if j!=I then

set nr\*=x-ax[i]

**STEP 10 :** Set dr\*=ax[i]-ax[j]

[End loop(j)]

**STEP 11 :** set y+=(nr/dr)\*dy[i]

[End of loop(i)]

**STEP 12 :** Print x, y as solution

**STEP 13 :** Exit.

**C++ Program:**

#include<iostream>

#include<iomanip>

#define MAX 100

Using namespace std;

int main()

{

float ax[MAX+1],ay[MAX+1],nr,dr,x,y=0;

int i,j,n;

cout<<"\n Name: Sahil Sao "<<"\n Class: MCA II"<<"\n Path: d: Sahil/LegIntr.cpp";

cout<<"Enter the value of n"<<endl;

cin>>n;

cout<<"Enter the set of values"<<endl;

for(i=0;i<=n;i++)

cin>>ax[i]>>ay[i];

cout<<"Enter the value of x for which"<<"value of y is wanted"<<endl;

cin>>x;

cout<<"fixed";

for(i=0;i<=n;i++)

{

nr=dr=1;

for(j=0;j<=n;j++)

{

nr=dr=1;

for(j=0;j<=n;j++)

if(j!=1)

{

nr\*=x-ax[j];

dr\*=ax[i]-ax[j];

}

y+=(nr/dr)\*ay[i];

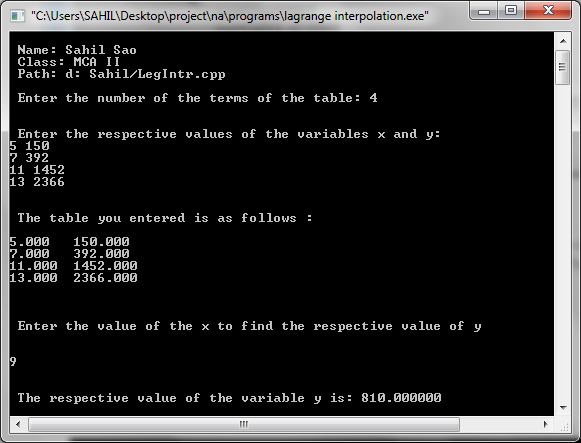
} cout<<"Whenx="<<setw(4)<<setprecision(1)<<x<<"y="<<setw(7)<<setprecision(1)<<y;

return 0;

}

}

**Output:**



**ASSIGNMENT-11**

**Objective: Write a Program to implement Newton’s backward Interpolation method.**

Start

Get the value of n

Get element of ax,ay

Get value of xelement of ax,ayion]

H=ax[1]-ax[0]

For i=0 to n-1

Diff[i][1]=ay[i+1]-ay[i]

Loop for j=2 to ORDER

I=i-1

P=(x-ax[i])/h

Yp=ay[i]

End Loop(i)

Loop for i=0 to n-j

Diff[i][j]=diff[i+1][j-1]-diff[i][j-1]

End Loop(i)

End Loop(j)

I=n

Is A x[i]>x

**Stop**

A

For K=1 to ORDER

Nr\*=p-k+1

Dr\*=K

Yp+=(nr/dr)\*diff[j][k]

End Loop K

Print x,y,p as solution

Stop

**Flow Chart:**

**Algorithm:**

**STEP 1 :** Start.

**STEP 2 :** Get the value of n.

**STEP 3 :** Get the elements of ax,ay.

**STEP 4 :** Get value of n.

**STEP 5 :** h=ax[1]-ax[0].

**STEP 6 :** for i=0 to n-i.

**STEP 7 :** diff[i][1]=ay[i+1]-ay[i].

[End of loop(i)]

**STEP 8 :** loop j=2 to codes.

**STEP 9 :** for i=0 to n-j.

**STEP 10 :** diff[i][j]=diff[i+1][j-1]-diff[i][j-1]

[End loop(i)]

[End loop(j)]

**STEP 11 :** set i=0.

**STEP 12 :** if ax[i]>x then.

(a)i=i+1.

(b) else

I=i-1

P=(x-ax[i]/h)

Yp=ay[i]

For k=1 to order

nr\*=p-k+1

dr\*=k

yp++(nr/dr)\*diff[i][k]

[End loop k]

**STEP 13 :** print x,y,pas solution.

**STEP 14 :** Stop

**C++ Program:**

#include<iostream.h>

#include<iomanip.h>

#define MAXN 100

#define ORDER 4

int main()

{

float ax[MAXN+1],ay[MAXN+1],

diff[MAXN+1][ORDER+1],

nr=1.0,dr=1.0,x,p,h,yp;

int n,i,j,k;

cout<<"Name: Sahil Sao"<<"\nclass: MCA II"<<"\npath: d:/Sahil/NewBackInt.cpp";

cout<<"Enter the value of n"<<endl;

cin>>n;

cout<<"Enter the values in form x,y"<<endl;

for(i=0;i<=n;i++)

cin>>ax[i]>>ay[i];

cout<<"Enter the values of x" <<"for which value of y is wanted"<<endl;

cin>>x;

cout<<"fixed";

h=ax[1]-ax[0];

for(i=0;i<=n-1;i++)

diff[i][1]=ay[i+1]-ay[i];

for(j=2;j<=ORDER;j++)

for(i=0;i<=n-j;i++)

diff[i][j]=diff[i+1][j-1]-diff[i][j-1];

i=0;

while(!(ax[i]>x))i++;

i--;

p=(x-ax[i])/h;

yp=ay[i];

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

{

nr\*=p-k+1;

dr\*=k;

yp+=(nr/dr)\*diff[i][k];

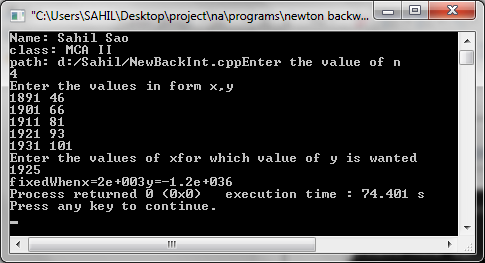
}

cout<<"Whenx="<<setw(6)<<setprecision(1)<<x<<"y="<<setw(6)<<setprecision(2)<<yp;

return 0;

}

**Output:**



**ASSIGNMENT-12**

**Objective: Write a Program to implement Newton forward interpolation method.**

**Flow Chart:**

Is ax[i]>x

40

30

i=i+1

30

i=i-1

P=(x-ax[i])/h

yp=ay[i]

For k=1 to ORDER

nr\*=p-k+1

dr\*=k

yp+=(nr/dr)\*diff[i][k]

End Loop (k)

Print x,y,p as solution

For i=0 to n-1

h=ax[1]-ax[0]

diff[i][1]=ay[i+1]-ay[i]

Get the value of n

Get elements of ax, ay

Get value of x

End Loop(i)

Loop for j=2 to ORDER

Loop for i=0 to n-j

diff[i][j]=diff[i+1][j-1]-diff[i][j-1]

End Loop (i)

End Loop (j)

i=0

**Algorithm:**

**STEP 1 :** Start.

**STEP 2 :** Input n.

**STEP 3 :** Input ax,ay

**STEP 4 :** Input x.

**STEP 5 :** Set h=ax[i]-ax[0].

**STEP 6 :** Loop for i=0 to n-1.

**STEP 7 :** diff[i][1]=ay[i+1]-ay[i]

[End loop(i)]

**STEP 8 :** Loop for j=2 to ORDER

**STEP 9 :** Loop for i=0 to n-j.

**STEP 10 :** Set diff[i][1]=ay diff[i+1][j-1]-diff[i][j-1]

[End loop(i)]

[End loop(j)]

**STEP 11 :** Set i=0.

**STEP 12 :** If ax[i]>x

**STEP 13 :** Set i=i+1

Repeat step(12)&(13)

Else

**STEP 14 :** set i=i-1.

**STEP 15 :** p=(x-ax[i])/h

**STEP 16 :** yp=ay[i]

**STEP 17 :** for k=1 to ORDER.

**STEP 18 :** set nr\*=p-k+1.

**STEP 19 :** set dr\*=k.

**STEP 20 :** set yp+=(nr/dr)\*diff[i][k]

[End of loop]

**STEP 21 :** Print x,y,p as solution.

**STEP 22 :** Exit.

**C++ Program:**

#include<iostream>

#include<iomanip>

#define MAXN 100

#define ORDER 4

using namespace std;

int main()

{

float ax[MAXN+1],ay[MAXN+1],

diff[MAXN+1][ORDER+1],

nr=1.0,dr=1.0,x,p,h,yp;

int n,i,j,k;

cout<<"Name: Sahil Sao"<<"\nclass: MCA II"<<"\npath: d:/Sahil/NewFarInt.cpp";

cout<<"Enter the value of n"<<endl;

cin>>n;

cout<<"Enter the values in form x,y"<<endl;

for(i=0;i<=n;i++)

cin>>ax[i]>>ay[i];

cout<<"Enter the values of x"<<"for which value of y is wanted"<<endl;

cin>>x;

cout<<"fixed";

h=ax[1]-ax[0];

/\* now making the diff. table \*/

/\* Calculating the first order differences \*/

for(i=0;i<=n-1;i++)

diff[i][1]=ay[i+1]-ay[i];

/\* Calculating the second & higher order differences \*/

for(j=2;j<=ORDER;j++)

for(i=0;i<=n-j;i++)

diff[i][j]=diff[i+1][j-1]-diff[i][j-1];

/\* now finding x0 \*/

i=0;

while(!(ax[i]>x))i++;

/\* now ax[i] is x0 & ay[i] is y0 \*/

i--;

p=(x-ax[i])/h;

yp=ay[i];

/\* Now carrying out interpolation \*/

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

{

nr\*=p-k+1;

dr\*=k;

yp+=(nr/dr)\*diff[i][k];

}

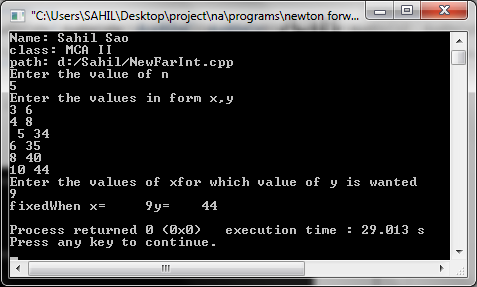
cout<<"When x="<<setw(6)<<setprecision(1)<<x<<"y="<<setw(6)<<setprecision(2)

<<yp<<endl;

return 0;

}

**Output:**



**ASSIGNMENT-13**

**Objective: Write a Program to implement Numerical differencing using backward method.**

**Flow Chart:**

inputn

input x

Print h

For i=0 to n

Print x1, y1

for i=0 to n

z = i

c[i] = 0.0

For j=0 to n-1 to step 1

For j=0 to n-1

End of i

End for i

find for j

c[j] = b[z]

i = n+1

b[i] = b[i] – b[i-1]

For i=2 to n-1

End for i

End for i

input a[i]

if

a[i] = = x1

1

y1 = y1/h

End for i

y1 = y1 + c[i] /(i+1)

For i=0 to n-1

h = a[1] – a[0]

**Algorithm:**

**STEP 1 :** Variable initialization.

**STEP 2 :** input n

**STEP 3 :** Input for x

for loop i=0 to n

print a [i]

set e[0][i] = a[i]

End of for loop

**STEP 4 :** Input for y

for loop i=0 to n

print b[i]

set e[1][i] = b[i]

(End of for loop)

**STEP 5 :** [Enter the value of x at which y is to be found (Backword)] print x1.

**STEP 6 :** Repeat to step 7 for loop i=0 to n.

**STEP 7 :** If (a[i]= = x1)

set z=i.

Break.

(end of if)

**STEP 8 :** Repeat to step 9 for loop j=0 to n-1

**STEP 9 :** For loop (i = n-1;i>0;i--)

set b[i] = b[i] – b[i-1]

(End of loop)

set ([j]=b[z])(End of step 8 for loop)

**STEP 10 :** For loop i =2 to n-1

set c[i] = 0.0

h= a[1]-a[0]

**STEP 11 :** Print h.

**STEP 12 :** For loop i = 0 to n-1

set y1 += c[i]/(i+1)

set y1 = y1/n.

**STEP 13 :** Print x1, y1.

**STEP 14 :** Exit.

**C++ Program:**

#include <iostream>

#include<conio.h>

#include<math.h>

#include<iomanip>

using namespace std;

int main(){

float a[20], b[20], c[20], e[20][20],x1, h, y1 =0.0;

int i,n,j,z,k,w;

clrscr();

cout<<"Name: Sahil Sao"<<"\nclass: MCA II"<<"\npath: d:/Sahil/NumerDiff.cpp";

cout<<"Enter the number of terms";

cin>> n;

cout<<"\n Enter the value of x \n";

for(i=0;i<n;i++)

{

cout<<"x ["<< i+1<<"] = ";

cin>>a[i];

}

cout<<"\n Enter the value of y \n";

for (i=0;i<n;i++)

{

cout<<"y["<< i+1<<"] =";

cin>> b[i];

e[1][i] = b[i];

}

cout<<"\n Enter the value of x for computing";

cin>>x1;

for(i=0;i<n;i++)

{

if(a[i]==x1)

{

z= i;

break;

}

}

for (j=0;j<n-1;j++)

{

for(i=n-1;i>0;i--)

{

b[i] = b[i] \*b[i-1];

}

c[j] = b[z];

}

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

c[i] = 0.0;

h=a[1] \*a[0];

cout<<"h = "<<h;

for(i=0;i<n-1;i++)

y1 += c[i] / (i+1);

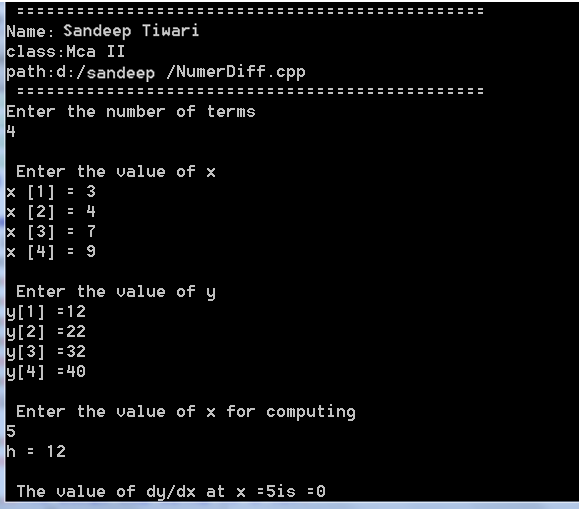
y1 = y1/h;

cout<<"\n\n The value of dy/dx at x ="<<x1<<"is ="<<y1;

getch();

}

**Output:**



**ASSIGNMENT-14**

**Objective: Write a program to implement Trapezoidal rules.**

**Flow Chart:**

Define function y(x)

Print (h/2)\*s as Solution

S=y(x0)+y(xn)

h=(xn-x0)/n

Loop for i=1, n-1

Get values of x0,xn,n

S+=2\*y(x0+i\*h)

End Loop (i)

**Algorithm:**

**STEP 1 :** Start.

**STEP 2 :** Define function y(x).

**STEP 3 :** Input x0,xn,n.

**STEP 4 :** Set n=(xn-x0)/n.

**STEP 5 :** Set s=y(x0)+y(xn).

**STEP 6 :** Loop for i=1 to n-1.

**STEP 7 :** Set st=2\*y(x0+i\*h).

[End loop(i)]

**STEP 8 :** Print (n/2)\*s as solution.

**STEP 9 :** Exit.

**C++ Program:**

#include<iostream>

#include<iomanip>

Using namespace std;

float y(float x)

{

return 1/(1+x\*x);

}

int main()

{

float x0,xn,h,s;

int i,n;

cout<<"Name: Sahil Sao"<<"\nclass: MCA II"<<"\path:D:/Sahil/Trapozoidl.cpp";

cout<<"enter x0,xn no. of subintervals"<<endl;

cin>>x0>>xn>>n;

h=(xn-x0)/n;

s=y(x0)+y(xn);

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

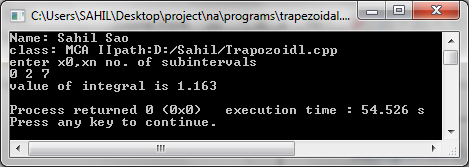
s +=2\*y(x0+i\*h);

cout<<"value of integral is"<<setw(6)<<setprecision(4)<<(h/2)\*s<<endl;

return 0;

}

**Output:**



**ASSIGNMENT-15**

**Objective: Write a program to implement Simpsons 1/3 rules.**

**Flow Chart:**

Define function f(x)

Print (h/3)\*s as Solution

S=y0+yn+4y1

h=(xn-x0)/n

Loop for i=3 to n-1 Step 2

Get values of x0,xn,n

S+=4\*yi+2\*yi-1

End Loop (i)

**Algorithm:**

**STEP 1:** variable inilitization.

**STEP 2:** input the value of x0,xn,n.

**STEP 3:** print x0,xn,n.

**STEP 4:** set lenh = (xn – x0)/n.

**STEP 5:** set s = y(x0) +y(xn) + 4\*y(x0+h)

**STEP 6:** for loop (i=3;i<=n-1;i+=2)

**STEP 7:** set s+= 4\*y(x0 + i \* lenh) + 2\*y (x0+ (i-1) \*h)

**STEP 8:** print(lenh/3)\*s.

**STEP 9:** Exit.

**C++ Program:**

#include<iostream.h>

#include<iomanip.h>

float y(float x)

{

return 1/(1+x\*x);

}

int main()

{

float x0,xn,h,s;

int i,n;

cout<<"Name: Sahil Sao"<<"\nclass:MCA II"<<"\path:D:/Sahil/Simp1/3.cpp";

cout<<”Enter x0,xn .no. of subintervals”<<endl;

cin>>x0>>xn>>n;

h=(xn-x0)/n;

s=y(x0)+y(xn)+4\*y(x0+h);

for(i=3;i<=n-1;i+=2)

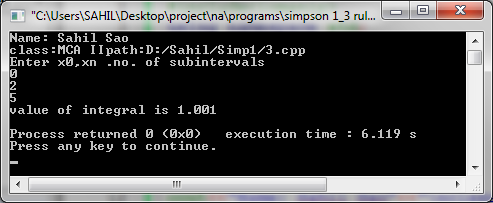
s +=4\*y(x0+i\*h)+2\*y(x0+(i-1)\*h);

cout<<”value of integral is”<<setw(6)<<setprecision(4)<<(h/3)\*s<<endl;

return 0;

}

**Output:**



**ASSIGNMENT-16**

**Objective: Write a program to implement Simpsons 3/8 rules.**

**Flow Chart:**

Define Function y(x)

Z value of x0,xn,n

Lenh=(xn-x0)/n

S1=y(x0)+y(xn)

For i=1;i<n-1;i+=3;

S2=s2+y(x0)+i\*lenh+y(x0)+(i+1)\*lenh

i=3 to n-3,i+=3

S3==s3++y(x0)+i\*lenh

S=((3\*lenh)y8)\*(s1\*s1+3\*s2+2\*s3)

Print S

**Algorithm:**

**STEP 1:** Variable initialization.

**STEP 2:** Input the value of x0, xn,n.

**STEP 3:** print x0,xn,n.

**STEP 4:** set lenh = (xn – x0) /n.

**STEP 5:** set s1 = y(x0) + y(xn)

**STEP 6:** for loop (i=3;i<=n-3;i+=3)

**STEP 7:** set s3 = s3 + y (x0 + i\*lenh) \* y(x0 + (i+1) \* lenh)

**STEP 8:** For loop (i=3; i<=n-3;i+=3)

**STEP 9:** set s2 = s2 + y(x0 + i \* lenh)

**STEP 10:** set s = ((3 \* lenh)/8) \* (s1 + 3 \* s3 + 2 \*s2)

**STEP 11:** print s

**STEP 12:** Exit.

**C++ Program:**

#include<iostream.h>

#include<conio.h>

#include<math.h>

#include<stdio.h>

#include<iomanip.h>

float y(float x)

{

return(1/(1+x\*x));

}

void main()

{

float x0,xn,s,s1,s2,s3;

float lenh;

int i,n;

//char ch;

//clrscr();

cout<<"Name:Sahil Sao"<<"\nclass:MCA II"<<"\path:D:/Sahil/Simp3/8.cpp";

cout<<"\nEnter the x0,xn,No. of Subinterval"<<endl;

cin>>x0>>xn>>n;

lenh=(xn-x0)/n;

cout<<"\nSimpson's 3/8 Rule";

s1=y(x0)+y(xn);

for(i=1;i<=n-1;i+=3)

s2=s2+y(x0+i\*lenh)+y(x0+(i+lenh)\*lenh);

for(i=3;i<=n-3;i=i+3)

s3=s3+y(x0+i\*lenh);

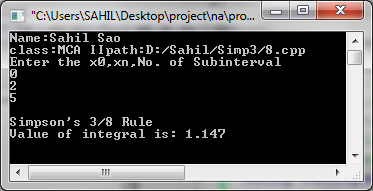
s=((3\*lenh)/8)\*(s1+3\*s2+2\*s3);

cout<<"\nValue of integral is:"<<setw(6)<<setprecision(4)<<s<<endl;

getch();

}

**Output:**



**ASSIGNMENT-17**

**Objective: Write a Program to implement Euler Modified method.**

**Flow Chart:**

Start

Define function df(x,y)

Define function ms(y,y1)

X1=x0.y1=y0

yy=y0 yp=yy

Yp=y1 y1=yy+0.1\*(ms(yy,df(x1,y1)))

y

err= y1-yp

Y1=yy+0.1\*df(x1,y1)

Yy=y1 x1+=h

Get values of x0,v0,h,x

Is x1>x

10

Stop

Print x1.v1

Is x1=x0

X1=x1+h

Is err<.0001

10

**Algorithm:**

**STEP 1 :** Start.

**STEP 2 :** Define function df(x,y).

**STEP 3 :** Define function ms(y,y1).

**STEP 4 :** Get values of x0,y0,h,x

yy=y0

yp=yy

x1=x0

y1=y0

**STEP 5 :** If x1>x then

(a)Stop

else

yp=y1;

**STEP 6 :** y1=yy+0.1\*(ms(yy,dy(x1,y1)))

Print x1,y1.

**STEP 7 :** If (err<.0001)

y1=yy+0.1\*df(x1,y1)

yy=y1

x1+=h

go to step 6.

**C++ Program:**

#include<stdio.h>

#include<math.h>

#include<conio.h>

void main()

{

float x, y, x1 =0.0, y1 =0.0, h, ms = 0.0, flag = 0, y2 = 0.0, t=0.0;

int i, j;

clrscr();

printf(“Name:Sahil Sao");

printf(“\nclass:MCA II”);

printf("\npath:D:/Sahil/Euler.cpp");

printf(“\n Enter the value of x”);

scanf(“%f”, &x);

printf(“Enter the value of y”);

scanf(“%f”, &y);

printf(“Enter the height”);

scanf(“%f”, &h);

i = 7;

printf(“x”); gotoxy(10,i); printf(“x+y=y1”); gotoxy(28,i);

printf(“mean slope”);

gotoxy(45,i); printf(“old y+.1(mean slope) = new y”);

while(x1<x)

{

i++;

do

{

i++

if(flag = =0)

{

y1 = x1+y;

gotoxy(2,i); printf(“%.1f”, x1); gotoxy (10,i); printf(“%.5f”,y1);

gotoxy(28,i);

printf(“%.5f”, ms);

m5 = y1;

y2 = y+h\*ms;

gotoxy(45,i); printf(“%.5f”, y2);

x1 = x1+h;

flag = 1;

}

else

{

ms = (y1 +(x1 + y2))/2.0;

t = y+h\*ms;

if (y2 = = t)

{

y2 = y+h\*ms;

break;

}

goto(2,i);

printf(“%d.1f”,x1); gotoxy(10,i); printf(“%.1f+%.5f”, x1 ,y2); y2 = y+h\*ms;

gotoxy(28,i);

printf(“%.5f” ms);

gotoxy(45,i);

printf(“%.5f”,y2);

}

} while(1);

y = y2;

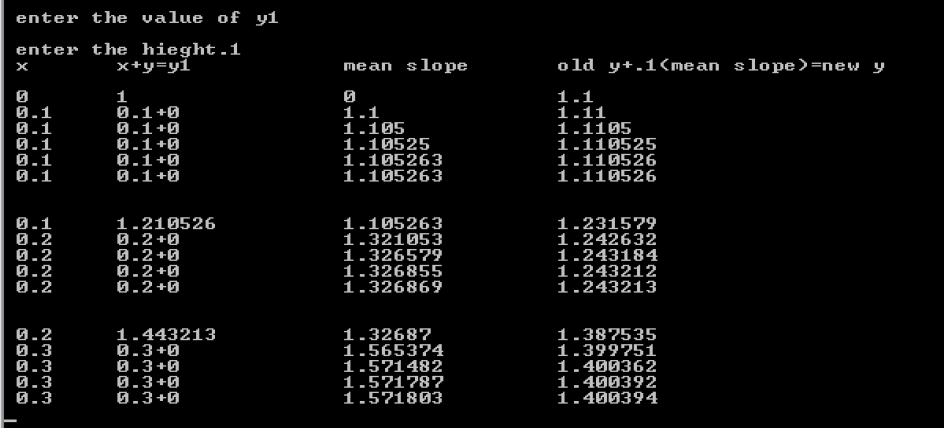
printf(“\n\n”);

flag = 0;

}

}

**Output:**

****

**ASSIGNMENT-18**

**Objective: Write a Program to implement Euler method.**

**Flow Chart:**

Define function df(X,Y)

Get values of x0,y0,h,x

y1=Y0

y1+=h\*df(x1,y1)

x1+=+h

Print X1,y1

10

10

Is x1>x

x1=x0

**Algorithm:**

**STEP 1 :** Start.

**STEP 2 :** Define function df (x,y).

**STEP 3 :** Input x0,y0,h,x.

**STEP 4 :** Set x1=y0.

**STEP 5 :** Set y1=x0

Repeat step(6) to (9) till x1>x

**STEP 6 :** If x1>x

Then return.

**STEP 7 :** else

Set y1+h\*df(x1,y1)

**STEP 8 :** Set x1+=+h

**STEP 9 :** set x1,y1

**STEP 10 :** Exit.

**C++ Program:**

#include<iostream.h>

#include<iomanip.h>

#include<conio.h>

double df(double x, double y)

{

return x\*y;

}

void main()

{

clrscr();

double x0,y0,h,x,y=0,oldy;

cout<<"Name: Sahil Sao"<<"\nclass: MCA II"<<"\npath: d:/Sahil/Eular.cpp";

cout<<"\nEnter the values of x0, y0, h, x"<<endl;

cin>>x0>>y0;

cout<<"\nEnter the values h "<<endl;

cin>>h;

cout<<"\nEnter the values of x "<<endl;

cin>>x;

//cout<<fixed;

oldy=y0;

while(x0<x)

{

y = oldy+(h\*df(x0,y0));

oldy=y;

x0=x0+h;

cout<<x0<<setw(10)<<df(x0,y0)<<setw(10)<<y<<endl;

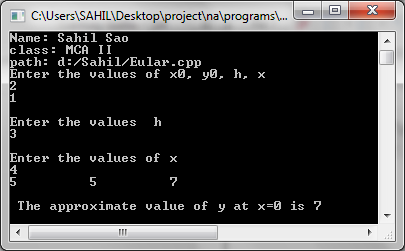
}

cout<<"\n The approximate value of y at x=0 is "<<y<<endl;

getch();

}

**Output:**



**ASSIGNMENT-19**

**Objective: Write a Program to implement Runge kutta 3rd order method.**

**Flow Chart:**

Define function f(x,y)

Get values of x0,y0,h,xn

K1=h\*f(x,y)

K2=h\*f(x+h/2,y+k1/2)

K3=h\*f(x+h/2,y+k2/2)

K4=h\*f(x+h,y+k3)

K=(k1+(k2+k3)\*2+k4)6

X=y+h

Y=y+k

Print x,y

Is x=xn?

20

20

**Algorithm:**

**STEP 1 :** Start.

**STEP 2 :** Define function f(x,y).

**STEP 3 :** Get value of x0,y0,h,xn.

**STEP 4 :** If x=xn then

(a)Stop

(b)else

K1=h\*f(x,y)

K2=h\*f(x+h/2,y+K1/2)

Kn=h\*f(x+h,y+K1)

K3=h\*f(x+h,y+Kn)

K=(K1+4K2+K5)/6

x=x+h

y=y+K.

**STEP 5 :** Print x,y.

**STEP 6 :** Goto step(4).

**C++ Program:**

#include<iostream.h>

#include<iomanip.h>

float f(float x,float y)

{

return x+y\*y;

}

int main()

{

float x0,y0,h,xn,x,y,k1,k2,k3,k4,k;

cout<<"Name:Sahil Sao"<<"\nclass:MCA II"<<"\path:D:/Sahil/RGK.cpp";

cout<<"Enter the values of x0,y0,"<<"h,xn"<<endl;

cin>>x0>>y0>>h>>xn;

x=x0;

y=y0;

cout<<"fixed";

while(1)

{

if(x==xn)break;

k1=h\*f(x,y);

k2=h\*f(x+h/2,y+k1/2);

k3=h\*f(x+h/2,y+k2/2);

k4=h\*f(x+h,y+k3);

k=(k1+(k2+k3)\*2+k4)/6;

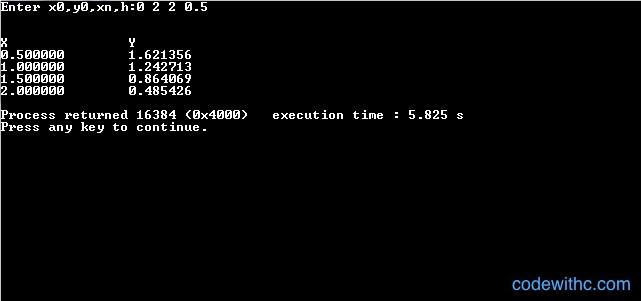
x+=h;y+=k; cout<<"Whenx="<<setprecision(4)<<setw(8)<<x<<"y="<<setw(8)<<y<<end;

}

return 0;

}

**Output:**



**ASSIGNMENT-20**

**Objective: Write a Program to implement miles predictor character method.**

Calculate starting values of x

Get values of 0,xr,h,aerr

Define function correct

Get starting values of y

Isx(3)>=xr?

20

X[4]=x[3]+h

Predict y(4)

Print x(4),y(4),f(4)

1

**Flow Chart:**

1

Call correct

Y(c)=y(4)

Call correct

Is fabs(yc-y[4])<aerr?

Prepare for next iteration

20

c

Calculate corrected y(4)

Print y(4),f(4)

30

30

**Algorithm:**

**STEP 1 :** Start

**STEP 2 :** Define function correct

**STEP 3 :** Input x0,xr,h,aerr

**STEP 4 :** Input y

**STEP 5 :** Calculate starting values of x

Repeat step (6) to(4)

**STEP 6 :** If x(3)>=xr

then Exit

**STEP 7 :** else

x[4]=x[3]+h

**STEP 8 :** predict y[4]

**STEP 9 :** print x(4),y(4),f(4)

**STEP 10 :** Call correct.

(a) calculate corrected y(4)

(b) print y(4),f(4)

(c) Stop

**STEP 11 :** set yc=y(4)

**STEP 12 :** call correct

**STEP 13 :** If fabs(yc-y[4])<aerr

Then prepare for next iteration

**STEP 14 :** Else

**STEP 15 :** Exit.

**C++ Program:**

#include<iostream.h>

#include<iomanip.h>

#include<math.h>

#include<conio.h>

float x[5],y[5],h;

float f(int i)

{

return x[i]-y[i]\*y[i];

}

void corect()

{

y[4] = y[2]+(h/3)\*(f(2)+4\*f(3)+f(4));

cout<<setw(23)<< ""<<setprecision(4)<<setw(8)<<y[4]<<setw(8)<<f(4)<<endl;

}

int main()

{

clrscr();

float xr,aerr,yc;

int i;

cout<<"\n Name:Sahil Sao"<<"\n Class:MCA II"<<"\n Path:d:/Sahil/predictor.cpp";

cout<<"Enter the values of x0,xr,h,"<<"allowed error"<<endl;

cin>>x[0]>>xr>>h>>aerr;

cout<<"Enter the value of y[i],i=0,3"<<endl;

for(i=1;i<=3;i++)

cin>>y[i];

for(i=1;i<=3;i++)

x[i]=x[0]+i\*h;

cout<<setw(5)<< "x" <<setw(15)<< "Predicted" <<setw(17)<< "Corrected" <<endl;

cout<<setw(11)<< "y" <<setw(10)<< "f" <<setw(7)<< "y" <<setw(10)<< "f" <<endl;

while(1)

{

if(x[3]>=xr)return 0;

x[4]=x[3]+h;

y[4]=y[0]+(4\*h/3)\*(2\*(f(1)+f(3))-f(2));

cout<<setw(6)<<setprecision(2)<< x[4] <<setprecision(4)<<setw(8)<< y[4]

<<setw(8)<< f(4) <<endl;

corect();

while(1)

{

yc=y[4];

corect();

if(fabs(yc-y[4])<=aerr)break;

}

for(i=0;i<=3;i++)

{

x[i]=x[i+1];

y[i]=y[i+1];

}

}

}

**Output:**

