1. Matrix Multiplication :

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
int main()
{
   int i,j,k,m,n,p,q;
   float a[10][10],b[10][10],c[10][10];
   fstream R("mat.dat");
   R>>n;
   R>>m;
   for(i=0;i<n;i++)
   {
      for(j=0;j<m;j++)
      {
         R>>a[i][j];
      }
   }
   R>>p;
   R>>q;
   for(i=0;i<p;i++)
   {
      for(j=0;j<q;j++)
      {
         R>>b[i][j];
      }
   }
   if(m==p)
   {
      for(i=0;i<n;i++)
      {
         for(j=0;j<q;j++)
         {
            c[i][j]=0;
            for(k=0;k<m;k++)
            {
               c[i][j]=c[i][j]+a[i][k]*b[k][j];
            }
            cout<<c[i][j]<<" ";
         }
         cout<<endl;
      }
   }
   else
   {
      cout<<endl<<"Matrix multiplication is not possible";
   }
}
```

2. Assending and Decending order sorting of a given data :

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
int main()
{
   int i,j,n;
   float a[50],c;
   fstream R("mat.dat");
   R>>n;
   for(i=0;i<n;i++)
   {
      R>>a[i];
   }
   for(i=0;i<n-1;i++)
   {
      for(j=i+1;j<n;j++)
      {
         if(a[i]>a[j])
         {
            c=a[i];
            a[i]=a[j];
            a[j]=c;
         }
      }
   }
   cout<<"\n in accending order : ";
   for(i=0;i<n;i++)
   {
      cout<<a[i]<<" ";
   }
   cout<<"\n in descending order : ";
   for(i=0;i<n;i++)
   {
      cout<<a[n-(i+1)]<<" ";
   }
}
```

3. Maximum and minimum , mean and standard daviation of a given data :

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
int main()
{
    int i,n;
    float a[50],c,max,min,sd,variance,sum=0,sum2=0,mean;
    fstream R("mat.dat");
    R>>n;
    for(i=0;i<n;i++)
    {
       R>>a[i];
    }
    max=a[0];
    min=a[0];
    for(i=0;i<n;i++)
    {
       if(a[i]>max)
       {
          max=a[i];
       }
       if(a[i]<min)
       {
          min=a[i];
       }
       sum=sum+a[i];
       sum2=sum2+a[i]*a[i];
    }
    cout<<"\n maximum No. is : "<<max;
    cout<<"\n minimum No. is : "<<min;
    mean=sum/float(n);
    cout<<"\n mean is : "<<mean;
    variance=sum2/n-mean*mean;
    sd=sqrt(variance);
    cout<<"\n standard daviation is : "<<sd;
}
```

4. Least Square Fit about a line :

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
int main()
{
    fstream R("datafit.dat");
    int i,j,k,n;
    //cout<<"\n enter the no. of data pairs to be entered: \n";
    R>>n;
    float x[n],y[n],a,b;
    //cout<<"\n enter the x-axis values\n";
    for(i=0;i<n;i++)
       R>>x[i];
    //cout<<"\n enter the y-axis values";
    for(i=0;i<n;i++)
        R>>y[i];
    float xsum=0,ysum=0,x2sum=0,xysum=0;
    for(i=0;i<n;i++)
    {
        xsum=xsum+x[i];
        ysum=ysum+y[i];
        x2sum=x2sum+pow(x[i],2);
        xysum=xysum+x[i]*y[i];
    }
    a=(n*xysum-xsum*ysum)/(n*x2sum-xsum*xsum);
    b=(x2sum*ysum-xsum*xysum)/(x2sum*n-xsum*xsum);
    float yf[n];
    for(i=0;i<n;i++)
        yf[i]=a*x[i]+b;
    cout<<"S.no"<<"    "<<"x"<<"        "<<"y(given)"<<"        "<<"y(fitted)"<<endl;
    cout<<"------------------------------------------------\n";
    for(i=0;i<n;i++)
        cout<<i+1<<"."<<"      "<<x[i]<<"          "<<y[i]<<"           "<<yf[i]<<endl;
        cout<<"\nthe linear fit line is of the form:\n"<<a<<"x + "<<b<<endl;
}
```

5. Bisection Method (Finding roots of a given function) :

Here we take $f(x) = x^2 - 5x + 6$

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
float f(float x)
{
    return(x*x-5*x+6);
}
int main()
{
    float a,b,y,t;
    cout<<"enter the lower range\n";
    cin>>a;
    cout<<"enter the higher range\n";
    cin>>b;
    y=(a+b)/2;

    while(abs(f(y))>00000.1)
    {
        if(f(a)*f(b)<0)
        {
            y=(a+b)/2;
            if(f(a)*f(y)<0)
            {
               b=y;
               //t=abs(f(y));
            }
            else
            {
               a=y;
               //t=abs(f(y));
            }
         }
         else
         {
            cout<<"change the range";
            break;
         }
      }
      cout<<endl<<"root is "<<y;
   }
```

6. <u>Newton Rapsion Method (finding root):</u>

```
#include<iostream>
#include<cmath>
using namespace std;
float f(float x)
{
   return(x*x-5*x+6);
}
float fp(float x)
{
   return(2*x-5);
}
int main()
{
   float xn,x0;
   cout<<"enter the sheed value";
   cin>>x0;
   xn=x0-f(x0)/fp(x0);
   while(abs(xn-x0)>0.00001)
   {
      x0=xn;
      xn=x0-f(x0)/fp(x0);
   }
   cout<<endl<<"root is :"<<xn;
}
```

# Dual root by Newton Rapsion :

$$f(z) = 0$$

$$u(x,y) + iv(x,y) = 0$$
$$u(x,y) = 0 \quad v(x,y) = 0$$

Programme for $f(z) = z^3$

```
#include<iostream>
#include<fstream>
#include<cmath>
#include<iomanip>
using namespace std;
float u(float x,float y)
{
   return(x*x*x-3*x*y*y-1);
}
float v(float x,float y)
{
   return(-y*y*y+3*x*x*y);
}
float ux(float x,float y)
{
   return(3*x*x-3*y*y);
}
float uy(float x,float y)
{
   return(-6*x*y);
}
float vx(float x,float y)
{
   return(6*x*y);
}
float vy(float x,float y)
{
   return(-3*y*y+3*x*x);
}
int main()
{
   float x0,y0,delx,dely,x,y;
   cout<<"enter the gause value\n";
   cin>>x0>>y0;
   while((abs(u(x0,y0))+abs(v(x0,y0)))>0.0001)
   {
      delx=(v(x0,y0)*uy(x0,y0)-u(x0,y0)*vy(x0,y0))/(ux(x0,y0)*vy(x0,y0)-uy(x0,y0)*vx(x0,y0));
      dely=(u(x0,y0)*vx(x0,y0)-v(x0,y0)*ux(x0,y0))/(ux(x0,y0)*vy(x0,y0)-uy(x0,y0)*vx(x0,y0));
      x0=x0+delx;
      y0=y0+dely;
   }
   cout<<endl<<"roots are : x = "<<x0<<" y = "<<y0;

}
```

7. Trapazoidal Method for integration :

$$\int_a^b \left(x^2 + x + \frac{1}{6}\right) dx$$

```
#include<iostream>

#include<fstream>

#include<cmath>

using namespace std;

float f(float x)

{

   return(x*x+x+1.0/6);

}

float inte(float a,float b,float n)

{

   float in,h,i;

   h=(b-a)/n;

   in=f(a)+f(b);

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

   {


      in=in+2*f(a+i*h);


   }

   return(in*h/2);

}

int main()

{

   float a,b,n,in1,in2;

   cout<<"enter the lower limit\n";

   cin>>a;

   cout<<"enter the higher limit\n";

   cin>>b;
   n=2;

   do

   {

      in1=inte(a,b,n);

      n=n+2;

      in2=inte(a,b,n);

   }

   while(abs(in1-in2)>0.0001);

   cout<<"\nintegral is :"<<in2;

   cout<<"\n no.steps : "<<n;

}
```

8. **Simpsion 1/3 method for integration :**

```
#include<iostream>
#include<cmath>
using namespace std;
float f(float x)
{
   return(x*x+x+1.0/6);
}
float inte(float a,float b,float n)
{
   float h,s,i;
   h=(b-a)/(2*n);
   s=f(a)+f(b);
   for(i=1;i<=2*n-1;i=i+2)
   s=s+4*f(a+i*h);
   for(i=2;i<=2*n-2;i=i+2)
   s=s+2*f(a+i*h);
   return(s*h/3);
}
int main()
{
   float in1,in2,a,b,n;
   cout<<"enter the lower limit";
   cin>>a;
   cout<<"enter the higher limit";
   cin>>b;
   n=0;

   do
   {
      in1=inte(a,b,n);
```

```
        n=n+1;
        in2=inte(a,b,n);
    }
    while(abs(in1-in2)>0.00001);

    cout<<"\nintegral is :"<<in2;
    cout<<"\n no.steps : "<<n;
}
```

9. **Euler Method to solve Differential Equation :**

$$\frac{dy}{dx} = -y$$

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
float f(float x,float y)
{
    return(-y);
}
int main()
{
    float x0,y0,xf,h,x;
    fstream R("NR.dat");
    cout<<"enter the initial value : ";
    cin>>x0>>y0;
    cout<<"enter the final value of x : ";
    cin>>xf;
    cout<<"enter the fracttion i.e h : ";
    cin>>h;
    for(x=x0;x<xf;x=x+h)
    {
        R<<x<<"   "<<y0<<endl;
        y0=y0+h*f(x,y0);
    }
}
```

10. **Modified Euler Method :**

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
float f(float x,float y)
{
    return(-y);
}
int main()
{
    float x0,y0,xf,h,x;
    fstream R("NR.dat");
    cout<<"enter the initial value : ";
    cin>>x0>>y0;
    cout<<"enter the final value of x : ";
    cin>>xf;
    cout<<"enter the fracttion i.e h : ";
    cin>>h;
    for(x=x0;x<xf;x=x+h)
    {
        R<<x<<"   "<<y0<<endl;
        y0=y0+(h*f(x,y0)+h*f(x+h,y0+f(x,y0)))/2;
    }
}
```

11. **2nd order Runge Kutta method(solving DE) :**

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
float f(float x,float y)
{
    return(-y);
}
int main()
{
    float x0,y0,xf,h,x,k1,k2;
    fstream R(“NR.dat”);
    cout<<”enter the initial value : “;
    cin>>x0>>y0;
    cout<<”enter the final value of x : “;
    cin>>xf;
    cout<<”enter the 10raction i.e h : “;
    cin>>h;
    for(x=x0;x<xf;x=x+h)
    {
        R<<x<<”   “<<y0<<endl;
        k1=h*f(x,y0);
        k2=h*f(x+h,y0+k1);
        y0=y0+(k1+k2)/2;
    }
}
```

12. **4th order Runge Kutta :**

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
float f(float x,float y)
{
    return(-y);
}
```

```
int main()
{
   float x0,y0,xf,h,x,k1,k2,k3,k4;
   fstream R(“NR.dat”);
   cout<<”enter the initial value : “;
   cin>>x0>>y0;
   cout<<”enter the final value of x : “;
   cin>>xf;
   cout<<”enter the 11raction i.e h : “;
   cin>>h;
   for(x=x0;x<xf;x=x+h)
   {
      R<<x<<”   “<<y0<<endl;
      k1=h*f(x,y0);
      k2=h*f(x+h/2,y0+k1/2);
      k3=h*f(x+h/2,y0+k2/2);
      k4=h*f(x+h,y0+k3);
      y0=y0+(k1+k2+k3+k4)/6;
   }
}
```

13. Solving Second Order DE using Runge Kutta 4 :

$$\frac{d^2y}{dx^2} = f\left(x, y, \frac{dy}{dx}\right)$$

Initial condition –

$$y_0 = y(x_0),\ \ u_0 = \left(\frac{dy}{dx}\right)_{x_0}$$

Let $\frac{dy}{dx} = u$ then, $\frac{du}{dx} = f(x, y, u)$

Now we have two 1st order DE

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
float f2(float x,float y,float u)
{
   return(-y);
}
float f1(float x,float y,float u)
{
   return(u);
}
int main()
{
   float x0,y0,xf,h,x,k1,k2,k3,k4,u0,m1,m2,m3,m4;
   fstream R(“NR.dat”);
   cout<<”enter the initial value : “;
```

```
    cin>>x0>>y0>>u0;
    cout<<"enter the final value of x : ";
    cin>>xf;
    cout<<"enter the 12raction i.e h : ";
    cin>>h;
    for(x=x0;x<=xf;x=x+h)
    {
      R<<x<<"   "<<y0<<"  "<<u0<<endl;
      m1=h*f1(x,y0,u0);
      k1=h*f2(x,y0,u0);
      m2=h*f1(x+h/2,y0+m1/2,u0+k1/2);
      k2=h*f2(x+h/2,y0+m1/2,u0+k1/2);
      m3=h*f1(x+h/2,y0+m2/2,u0+k2/2);
      k3=h*f2(x+h/2,y0+m2/2,u0+k2/2);
      m4=h*f1(x+h,y0+m3,u0+k3);
      k4=h*f2(x+h,y0+m3,u0+k3);
      u0=u0+(k1+2*k2+2*k3+k4)/6;
      y0=y0+(m1+2*m2+2*m3+m4)/6;
    }
}
```

14. Solving a coupled DE :

$$\frac{dy_1}{dx} = \beta y_1 + (\alpha + \gamma)y_2$$

$$\frac{dy_2}{dx} = (\alpha - \gamma)y_1 - \beta y_2$$

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
float f2(float x,float y,float u)
{
  float a=0.1,b=1,g=1;
  return((a-g)*y-b*u);
}
float f1(float x,float y,float u)
{
  float a=0.1,b=1,g=1;
  return(b*y+(a+g)*u);
}
int main()
{
  float x0,y0,xf,h,x,k1,k2,k3,k4,u0,m1,m2,m3,m4;
  fstream R("NR.dat");
  cout<<"enter the initial value : ";
  cin>>x0>>y0>>u0;
  cout<<"enter the final value of x : ";
  cin>>xf;
  cout<<"enter the 13raction i.e h : ";
  cin>>h;
  for(x=x0;x<=xf;x=x+h)
  {
     R<<x<<"   "<<y0<<"  "<<u0<<endl;
     m1=h*f1(x,y0,u0);
     k1=h*f2(x,y0,u0);
     m2=h*f1(x+h/2,y0+m1/2,u0+k1/2);
     k2=h*f2(x+h/2,y0+m1/2,u0+k1/2);
     m3=h*f1(x+h/2,y0+m2/2,u0+k2/2);
     k3=h*f2(x+h/2,y0+m2/2,u0+k2/2);
     m4=h*f1(x+h,y0+m3,u0+k3);
     k4=h*f2(x+h,y0+m3,u0+k3);
     u0=u0+(k1+2*k2+2*k3+k4)/6;
     y0=y0+(m1+2*m2+2*m3+m4)/6;
  }
}
```

15. <u>Integration using Monte Carlo Method :</u>

$$I = \int_0^{\pi} \sin(x)\,dx$$

```
#include<iostream>
#include<fstream>
#include<cmath>
#include<cstdlib>
#include<ctime>
using namespace std;
float f(float x)
{
   return(sin(x));
}
int main()
{
   float a=0,b=M_PI,m=0,n1=0,n=10000,s,i,x,y;
   srand(time(NULL));
   for(x=a;x<=b;x=x+0.01)
   {
      if(f(x)>m)
      {
      m=f(x);
      }
   }
   for(i=1;i<=n;i++)
   {
      x=a+(b-a)*(rand()/float(RAND_MAX));
      y=m*(rand()/float(RAND_MAX));
```

```
        if(f(x)>y)
        {
            n1++;
        }
    }

    s=(b-a)*m*n1/float(n);
    cout<<endl<<”I =  “<<s;

}
```

$$I = \int_0^{\pi} cos(x)\,dx$$

```
#include<iostream>
#include<fstream>
#include<cmath>
#include<cstdlib>
#include<ctime>
using namespace std;
float f(float x)
{
    return(cos(x));
}
int main()
{
    float a=0,b=M_PI,m=0,n1=0,n=10000,s,i,x,y,min=0,n2=0;
    srand(time(NULL));
    for(x=a;x<=b;x=x+0.01)
    {
        if(f(x)>m)
        m=f(x);
        if(f(x)<min)
        min=f(x);
    }
    for(i=1;i<=n;i++)
    {
        x=a+(b-a)*(rand()/float(RAND_MAX));
        y=min+(m-min)*(rand()/float(RAND_MAX));
        if(f(x)>y && y>0)
        n1++;
        if(f(x)<y && y<0)
        n2++;
    }
    s=(b-a)*(m-min)*(n1-n2)/float(n);
    cout<<endl<<”I = “<<s;
}
```

16. Solving n linear equation :

$$\begin{gathered}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \\
. \\
. \\
\boldsymbol{a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n}
\end{gathered}$$

Then

$$x_i^{r+1} = \frac{b_i}{a_{ii}} - \sum_{j=0}^{i-1} \frac{a_{ij}}{a_{ii}} x_i^r - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_i^r$$

```
#include<iostream>
#include<fstream>
#include<cmath>
using namespace std;
int main()
{
   float a[10][10],b[10],x[10],y[10],s=1;
   int i,j,n;
   fstream R(“NR.dat”);
   R>>n;
   for(i=1;i<=n;i++)
   {
      for(j=1;j<=n;j++)
      {
         R>>a[i][j];
      }
   }
   for(i=1;i<=n;i++)
   {
      R>>b[i];
      y[i]=1;
   }
   while (s>0.0001)
   {
      s=0;
      for(i=1;i<=n;i++)
      {          x[i]=b[i]/a[i][i];
         for(j=1;j<i;j++)
         {
            x[i]=x[i]-a[i][j]*x[j]/a[i][i];
         }
         for(j=i+1;j<=n;j++)
         {
            x[i]=x[i]-a[i][j]*y[j]/a[i][i];

         }
         s=s+abs(x[i]-y[i]);
         y[i]=x[i];
```

```
        }

      }

      for(i=1;i<=n;i++)
      cout<<endl<<y[i]<<"   ";
   }
```

17. Curve Fitting about $y = ax^2 + b$:

$$a = \frac{n\sum x_i^2 y_i - \sum x_i^2 \sum y_i}{n\sum x_i^4 - \left(\sum x_i^2\right)^2}$$

$$b = \frac{\sum x_i^4 \sum y_i - \sum x_i^2 \sum x_i^2 y_i}{n\sum x_i^4 - \left(\sum x_i^2\right)^2}$$

```
#include<iostream>
#include<fstream>
#include<cmath>
#include<iomanip>
using namespace std;
int main()
{
   fstream R("datafit.dat");
   int i,j,k,n;
   //cout<<"\n enter the no. of data pairs to be entered: \n";
   R>>n;
   float x[n],y[n],a,b;
   //cout<<"\n enter the x-axis values\n";
   for(i=0;i<n;i++)
      R>>x[i];
   //cout<<"\n enter the y-axis values";
   for(i=0;i<n;i++)
      R>>y[i];
   float x4sum=0,ysum=0,x2sum=0,x2ysum=0;
   for(i=0;i<n;i++)
   {
      x4sum=x4sum+pow(x[i],4);
      ysum=ysum+y[i];
      x2sum=x2sum+pow(x[i],2);
      x2ysum=x2ysum+x[i]*x[i]*y[i];
   }
   a=(n*x2ysum-x2sum*ysum)/(n*x4sum-x2sum*x2sum);
   b=(x4sum*ysum-x2sum*x2ysum)/(x4sum*n-x2sum*x2sum);
   float yf[n];
   for(i=0;i<n;i++)
      yf[i]=a*x[i]*x[i]+b;
   cout<<"S.no"<<setw(5)<<"x"<<setw(19)<<"y(given)"<<setw(19)<<"y(fitted)"<<endl;
```

```
  cout<<”-----------------------------------------------\n”;
  for(i=0;i<n;i++)
    cout<<i+1<<”.”<<setw(8)<<x[i]<<setw(15)<<y[i]<<setw(18)<<yf[i]<<endl;
    cout<<”\nthe linear fit line is of the form:\n”<<a<<”x + “<<b<<endl;
}
```

18. Nth root of a complex number :

$$(x+iy)^{\frac{1}{n}}$$

$$x+iy=re^{i\theta}$$

Where $r=\sqrt{x^2+y^2}$ $\qquad$ $\theta = tan^{-1}\frac{y}{x}$

$$(x+iy)^{\frac{1}{n}} = r^{\frac{1}{n}}e^{i\left(\frac{\theta}{n}+\frac{2\pi m}{n}\right)}$$ where m = 0,1,2,…,n-1

$$= r^{\frac{1}{n}}\left[cos\left(\frac{\theta}{n}+\frac{2m\pi}{n}\right)+isin\left(\frac{\theta}{n}+\frac{2m\pi}{n}\right)\right]$$

```
#include<iostream>
#include<fstream>
#include<cmath>
#include<iomanip>
using namespace std;
int main()
{
  float x,y,n,m,a,r;
  cout<<"enter the real part of complex number  : ";
  cin>>x;
  cout<<"enter the imaginay part of complex number : ";
  cin>>y;
  cout<<"enter the value of n : ";
  cin>>n;
  r=sqrt(x*x+y*y);
  if(x>0)
  a=atan(y/x);
  else
  a=atan(y/x)+M_PI;
  cout<<"\n are : \n";
  for(m=0;m<n;m++)
  {
    cout<<pow(r,1.0/n)*cos(a/n+2*m*M_PI/n)<<" +
i("<<pow(r,1.0/n)*sin(a/n+2*m*M_PI/n)<<")\n";
  }
}
```

Read as a single line there is no line break

19. Fermat’s principle :

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
    float m,h1=10,h2=50,l=10,x,s1,s2,s,sini,sinr,r,mu=2.6;
    m=h1+mu*sqrt(h2*h2+l*l);
    for(x=0;x<=l;x=x+0.01)
    {
        s1=sqrt(x*x+h1*h1);
        s2=sqrt(h2*h2+(l-x)*(l-x));
        s=s1+mu*s2;
        if(s<m)
        {
            m=s;
            sini=x/s1;
            sinr=(l-x)/s2;
            r=sini/sinr;
        }
    }
    cout<<endl<<r;
}
```