```
#define NP maxpop
#define D maxdim
#define gen_max genmax
#define F factor
#define CR cross
#define inibound_l inib_l
#define inibound_h inib_h
/*----Constant for rnd_uni()----------------*/
#define IM1 2147483563
#define IM2 2147483399
#define AM  (1.0/IM1)
#define IMM1  (IM1-1)
#define IA1   40014
#define IA2   40692
#define IQ1   53668
#define IQ2   52774
#define IR1   12211
#define IR2   3791
#define NTAB  32
#define NDIV  (1+IMM1/NTAB)
#define EPS1   1.2e-7
#define RNMX   (1.0-EPS1)

#include<stdlib.h>
#include<stdio.h>
#include<time.h>
#include<math.h>
#include<conio.h>
#include<memory.h>

int strategy,genmax,maxdim,maxpop;
float inib_l,inib_h,factor,cross,cost[50],x1[50][10],x2[50][10];

float rnd_uni(long *);
void assignd(int D,float a[], float b[]);
float evaluate(float [],long *);

float evaluate(float tmp[],long *nfe)
    {     float cost;
         (*nfe)++;
    cost=pow((pow(tmp[0],2)+tmp[1]-11),2)+pow((tmp[0]+pow(tmp[1],2)-7),2);
        return cost;
    }

float rnd_uni(long *idum)
       {
      long j; long k;
      static long idum2=123456789;
      static long iy=0;static long iv[NTAB]; float temp;
      if(*idum<=0)
      {
         if(-(*idum)<1)    *idum=1;  else  *idum=-(*idum);   idum2=(*idum);
         for(j=NTAB+7;j>=0;j--)
         {
           k=(*idum)/IQ1;
           *idum=IA1*(*idum-k*IQ1)-k*IR1;
           if(*idum<0)      *idum+=IM1;
           if(j<NTAB)       iv[j]=*idum;
         }
         iy=iv[0];
       }
       k=(*idum)/IQ1;
       *idum=IA1*(*idum-k*IQ1)-k*IR1;
       if(*idum<0)    *idum+=IM1;
       k=idum2/IQ2;
       idum2=IA2*(idum2-k*IQ2)-k*IR2;
       if(idum2<0)         idum2+=IM2;
       j=iy/NDIV;      iy=iv[j]-idum2;   iv[j]=*idum;
       if(iy<1)    iy+=IMM1;
       if((temp=AM*iy)>RNMX)     return RNMX;
       else                      return temp;
     }



void assignd(int D,float a[], float b[])
```

```
   {
    int j;
    for(j=0;j<D;j++)
    {
    a[j]=b[j];
    }
   }


void main(int argc, char *argv[])

{
    int i,j,k,a,b,c,d,e,count=0,imin,seed;
    long nfe;
    float cost_trial,trial[3],costmin,bestit[3],best[3],cmax;
    clock_t start, end;

    FILE *fpin_ptr;

    if(argc!=2)
     {
        printf("\n Usage: De<input-file>\n");
        exit(1);
     }

/*--------------------------Read iput data-----------------------------*/

    fpin_ptr = fopen(argv[1],"r");
    if(fpin_ptr==NULL)
      {
        printf("\n Cannot open input file\n");
        exit(1);
      }

    fscanf(fpin_ptr,"%d",&strategy);
    fscanf(fpin_ptr,"%d",&genmax);
    fscanf(fpin_ptr,"%d",&maxdim);
    fscanf(fpin_ptr,"%d",&maxpop);
    fscanf(fpin_ptr,"%f",&inib_l);
    fscanf(fpin_ptr,"%f",&inib_h);
    fscanf(fpin_ptr,"%f",&factor);
    fscanf(fpin_ptr,"%f",&cross);
    fscanf(fpin_ptr,"%d",&seed);

    fclose(fpin_ptr);

 long  rnd_uni_init= -(long)seed;    nfe=0;
 start = clock();

for (i=0;i<NP;i++)
{
    for (j=0;j<D;j++)                             /* rand()/32768.0*/

      x1[i][j]=inibound_l + rnd_uni(&rnd_uni_init)*(inibound_h-inibound_l);

cost[i]= evaluate(x1[i], &nfe);

      /*    printf("x1=%f     x2=%f      cost=%f     ",x1[i][0],x1[i][1],cost[i]);
    getch();*/
}
    costmin=cost[0];
    imin=0;
for(i=1;i<NP;i++)
{
    if(cost[i]<costmin)
    {
       costmin=cost[i];
       imin=i;
    }
}
    assignd(D,best,x1[imin]);
    assignd(D,bestit,x1[imin]);
    /*printf("\nbest=%f\n",best);*/

while (count<gen_max)
 {
    count++;
    imin=0;

    for (i=0;i<NP;i++)
    {
        do a=int (rnd_uni(&rnd_uni_init)*NP); while (a==i);
      /*printf("a=%d   ",a);*/

        do b=int (rnd_uni(&rnd_uni_init)*NP); while (b==i || b==a);
      /*printf("\nb=%d   ",b);*/

        do c=int (rnd_uni(&rnd_uni_init)*NP); while (c==i || c==a || c==b);
      /*printf("\n c=%d",c); */

        do d=int (rnd_uni(&rnd_uni_init)*NP); while (d==i || d==a || d==b || d==c);


        do e=int (rnd_uni(&rnd_uni_init)*NP); while (e==i || e==a || e==b || e==c || e==d);


 /*-----------------de/rand/1/bin-------------------------*/
        if (strategy==1)
         {
        j=int (rnd_uni(&rnd_uni_init)*D);
             /*printf("     j=%d",j);
            getch(); */

        for (k=1;k<=D;k++)
        {
            if ((rnd_uni(&rnd_uni_init))<CR  || k==D)
            {
            trial[j]=x1[c][j]+F*(x1[a][j]-x1[b][j]);
            }
            else trial[j]=x1[i][j];

            /*printf("r1=%f ,trial[%d]=%f ,      ",r1,j,trial[j]);
            getch();*/
        j=(j+1)%D;

        }
         }
/*-------------------------DE/best/1/bin----------------------*/

 else if (strategy==2)
     {
            j=int (rnd_uni(&rnd_uni_init)*D);


        for (k=1;k<=D;k++)
        {
            if ((rnd_uni(&rnd_uni_init))<CR  || k==D)
            {
            trial[j]=bestit[j]+F*(x1[a][j]-x1[b][j]);
            }
            else trial[j]=x1[i][j];

           j=(j+1)%D;
        }
        }

/*-----------------de/best/2/bin----------------------*/
  else if (strategy==3)
      {
            assignd(D,trial,x1[i]);

            j=int (rnd_uni(&rnd_uni_init)*D);

        for (k=1;k<=D;k++)
        {
            if ((rnd_uni(&rnd_uni_init))<CR  || k==D)
            {
        trial[j]=bestit[j]+F*(x1[a][j]+x1[b][j]-x1[c][j]-x1[d][j]);
            }
            else trial[j]=x1[i][j];
            j=(j+1)%D;
         }
      }

/*------------------de/rand/2/bin-----------------------*/
 else if (strategy==4)
     {
            assignd(D,trial,x1[i]);
            j=int (rnd_uni(&rnd_uni_init)*D);

        for (k=1;k<=D;k++)
        {
            if ((rnd_uni(&rnd_uni_init))<CR  || k==D)
            {
         trial[j]=x1[e][j]+F*(x1[a][j]+x1[b][j]-x1[c][j]-x1[d][j]);
            }
            else trial[j]=x1[i][j];


           j=(j+1)%D;
        }
     }

/*------------------de/rand-to-best/1/bin-----------------------*/

  else if (strategy==5)
     {
               assignd(D,trial,x1[i]);

            j=int (rnd_uni(&rnd_uni_init)*D);

        for (k=1;k<=D;k++)
        {
            if ((rnd_uni(&rnd_uni_init))<CR  || k==D)
            {
            trial[j]=trial[j]+F*(bestit[j]-trial[j])+F*(x1[a][j]-x1[b][j]);
            }
            else trial[j]=x1[i][j];


           j=(j+1)%D;
        }
      }
/*------------------de/rand/1/exp-----------------------*/

   else if (strategy==6)
     {
        j=int (rnd_uni(&rnd_uni_init)*D);
        k=0;
        do
           {
             trial[j]=x1[c][j]+F*(x1[a][j]-x1[b][j]);

             j=(j+1)%D;
             k++;
           }
         while((rnd_uni(&rnd_uni_init))<CR && k<D);
     }

 /*------------------de/best/1/exp-----------------------*/

else if (strategy==7)
     {
        j=int (rnd_uni(&rnd_uni_init)*D);
        k=0;
        do
           {
             trial[j]=bestit[j]+F*(x1[a][j]-x1[b][j]);

             j=(j+1)%D;
             k++;
           }
           while((rnd_uni(&rnd_uni_init))<CR && k<D);
     }

/*------------------de/best/2/exp-----------------------*/
```

```
 else if (strategy==8)
      {

        assignd(D,trial,x1[i]);
        j=int (rnd_uni(&rnd_uni_init)*D);
        k=0;
       do
          {
        trial[j]=bestit[j]+F*(x1[a][j]+x1[b][j]-x1[c][j]-x1[d][j]);

             j=(j+1)%D;
             k++;
          }
          while((rnd_uni(&rnd_uni_init))<CR && k<D);
     }

/*------------------de/rand/2/exp-----------------------*/

else if (strategy==9)
     {

        assignd(D,trial,x1[i]);
        j=int (rnd_uni(&rnd_uni_init)*D);
        k=0;
        do
           {
        trial[j]=x1[e][j]+F*(x1[a][j]+x1[b][j]-x1[c][j]-x1[d][j]);

             j=(j+1)%D;
             k++;
           }
        while((rnd_uni(&rnd_uni_init))<CR && k<D);
     }

/*------------------de/rand-to-best/1/exp-----------------------*/

  else

      {
            assignd(D,trial,x1[i]);
            j=int (rnd_uni(&rnd_uni_init)*D);
            k=0;
        do
           {
          trial[j]=trial[j]+F*(bestit[j]-trial[j])+F*(x1[a][j]-x1[b][j]);

             j=(j+1)%D;
             k++;
           }
           while((rnd_uni(&rnd_uni_init))<CR && k<D);
     }


cost_trial=evaluate(trial, &nfe);

   /*  printf("\ntrialcost=%f ,  cost[%d]=%f   ",cost_trial,i,cost[i]);
       getch();*/


           if (cost_trial<=cost[i])
        {
            for (j=0;j<D;j++)
            x2[i][j]=trial[j];
            cost[i]=cost_trial;
           if(cost_trial<costmin)
           {
            costmin=cost_trial;
            imin=i;
            assignd(D,best,trial);
           }
        }
        else for (j=0;j<D;j++)
            x2[i][j]=x1[i][j];

      /*  printf("x1=%f     x2=%f    ",x2[i][0],x2[i][1]);
      getch();  */

   }   /*--------------------end of FOR loop after while---------*/

    assignd(D,bestit,best);
    for (i=0;i<NP;i++)
      {
        for (j=0;j<D;j++)
        x1[i][j]=x2[i][j];
      }

      cmax=cost[0];

       for (i=1;i<NP;i++)
    {  if(cost[i]>cmax)
      cmax=cost[i];
    }

      if((cmax-0.0)<=0.000001)
           break;

 }  /*--------end of while loop------------------------*/


    for(i=0;i<NP;i++)
     {
       printf("x1=%f   x2=%f   ",x1[i][0],x1[i][1]);
       printf("cost[%d]=%f    ",i,cost[i]);
     }
    printf("\ncmax=%f\n",cmax);
    printf("\ncount=%d\n",count);
    printf("\ncostmin=%f\n",costmin);
    printf("\n NFE=%ld\n",nfe);
    end = clock();
    printf("The time was: %f\n", (end - start) / CLK_TCK);

}  /*-------------end of main()-------------------------*/




      /* while (count<gen_max)
    for (i=0;i<NP;i++)
    {
        do a=rnd_uni()*NP; while (a==i);
        do b=rnd_uni()*NP; while (b==i || b==a);
        do c=rnd_uni()*NP; while (c==i || c==a || c==b);
        j=rnd_uni()*D;
        for (k=1;k<=D;k++)
        {
            if (rnd_uni() < CR || k==D)
            {
            trial[j]=x1[c][j]+F*(x1[a][j]-x1[b][j]);
            }
            else trial[j]=x1[i][j];
            j=(j+1)/D;
        }
        score=evaluate(trial);
        if (score<=cost[i])
        {
            for (j=0;j<D;j++)
            x2[i][j]=trial[j];
            cost[i]=score;
        }
        else for (j=0;j<D;j++)
            x2[i][j]=x1[i][j];
    }

    for (i=0;i<NP;i++)
    {
        for (j=0;j<D;j++)
        x1[i][j]=x2[i][j];
    }
    count++;
} */
```