# Appendice D - Listati dei codici C principali

9 giugno 2020

**Esempio di programma utilizzato per la generazione dei reticoli di Ising bidimensionali**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>



void Incipit(void);
int Lettura(void);
double Caso(void);
int Spin(void);
void Griglie(int*** Ising,int n,int N,int M);
void Temperature(double* T,int n);
void Bordo(int* Plus,int* Minus,int N);
void Bubblesort(double* T,int n);
void Metropolis(int*** Ising,int* Plus,int* Minus,double* T,int n,int N,int M, int Steps);
void MagnetizzazioneMedia(double* m,int*** Ising,int n,int N,int M);



int main(void){

//DICHIARAZIONE DELLE VARIABILI
int i,j,k,l;
double seed;
int n,Steps;
int N,M;
double* T;
int* Plus;
int* Minus;
int*** Ising;
double* m;
FILE* fp;
FILE* fd;
double t1,t2,DeltaT;

//ALLOCAZIONE DINAMICA DELLA MEMORIA
//temperatura
T=calloc(10000,sizeof(double));
if(T==NULL){
printf("L'allocazione dinamica della memoria per le temperature è fallita.\n");
exit(EXIT_FAILURE);
}
```

```
//bordo
Plus=calloc(100,sizeof(int));
if(Plus==NULL){
printf("L'allocazione dinamica della memoria per le condizioni a bordo è fallita.\n");
exit(EXIT_FAILURE);
}
Minus=calloc(100,sizeof(int));
if(Minus==NULL){
printf("L'allocazione dinamica della memoria per le condizioni a bordo è fallita.\n");
exit(EXIT_FAILURE);
}
//reticolo di Ising
Ising=calloc(10000,sizeof(int**));
if(Ising==NULL){
printf("L'allocazione dinamica della memoria per le configurazioni è fallita.\n");
exit(EXIT_FAILURE);
}
for(i=0;i<10000;i++){
Ising[i]=calloc(100,sizeof(int*));
if(Ising[i]==NULL){
printf("L'allocazione dinamica della memoria della dimensione lineare di Ising è fallita.\n'
exit(EXIT_FAILURE);
}
for(j=0;j<100;j++){
Ising[i][j]=calloc(100,sizeof(int));
if(Ising[i][j]==NULL){
printf("L'allocazione dinamica della memoria della dimensione lineare di Ising è fallita.\n'
exit(EXIT_FAILURE);
}
}
}
//magnetizzazione
m=calloc(10000,sizeof(double));
if(m==NULL){
printf("L'allocazione dinamica della memoria delle magnetizzazione media è fallita.\n");
exit(EXIT_FAILURE);
}

//APERTURA DEI FILE
fp=fopen("Andamento.dat", "w");
if(fp==NULL){
printf("L'apertura del file sull'andamento della magnetizzazione è fallita.\n");
exit(EXIT_FAILURE);
}
fd=fopen("Campioni.dat", "w");
if(fd==NULL){
```

```
printf("L'apertura del file sui reticoli di Ising è fallita.\n");
exit(EXIT_FAILURE);
}

//DICHIARAZIONE DEL RETICOLO DI ISING
N=25;
M=25;

//INIZIALIZZAZIONE DEL TEMPO
seed=time(0);
srand48(seed);

//ITERAZIONI DELL'ALGORITMO DI METROPOLIS
Steps=1000000;

//INTRODUZIONE
Incipit();

//CONDIZIONI AL BORDO PERIODICHE
Bordo(Plus,Minus,N);

//SELEZIONE DEL NUMERO DI CONFIGURAZIONI
n=Lettura();

//PRIMA VALUTAZIONE SUL TEMPO DI ESECUZIONE DEL PROGRAMMA
t1=time(0);

//GENERAZIONE E ORDINAMENTO DELLE TEMPERATURE
Temperature(T,n);
Bubblesort(T,n);

//GENERAZIONE DEI RETICOLI NON TERMALIZZATI
Griglie(Ising,n,N,M);

//TERMALIZZAZIONE DEI MODELLI DI ISING
Metropolis(Ising,Plus,Minus,T,n,N,M,Steps);

//CALCOLO DELLA MAGNETIZZAZIONE MEDIA DI CIASCUN RETICOLO
MagnetizzazioneMedia(m,Ising,n,N,M);

//STAMPA DELL'ANDAMENTO DELLA MAGNETIZZAZIONE IN FUNZIONE DELLA TEMPERATURA
for(i=0;i<n;i++)fprintf(fp,"%lf %lf\n", T[i],m[i]);
for(i=0;i<n;i++){
fprintf(fd,"%lf %lf\n", T[i],m[i]);
for(j=0;j<N;j++){
for(k=0;k<M;k++){
```

```
fprintf(fd,"%d ", Ising[i][j][k]);
}
fprintf(fd,"\n");
}
fprintf(fd,"\n");
}

//SECONDA VALUTAZIONE SUL TEMPO DI ESECUZIONE DEL PROGRAMMA
t2=time(0);
DeltaT=t2-t1;
printf("Tempo di elaborazione del programma è %lf.\n", DeltaT);

//FINE FORMALE DEL PROGRAMMA
free(T);
free(Ising);
free(Plus);
free(Minus);
free(m);
fclose(fp);
fclose(fd);
return 0;

}



void Incipit(void){

printf("Il programma genera un numero letto da tastieria di griglie di distribuzione isotrop
printf("\n");

}


int Lettura(void){

int rip,n;
rip=0;
do{
if(rip==0){
printf("Inserire il numero di configurazioni da generare: si ricorda che deve essere un nume
}else{
printf("Errore, il numero non rispetta le condizioni, il numero deve essere un numero intero
}
scanf("%d", &n);
rip++;
```

```
}while((n<0)||(n>10000));
printf("\n");
return n;

}


double Caso(void){

double r;
r=(double)lrand48()/RAND_MAX;
return r;

}


int Spin(void){

double a,s;
a=Caso();
if(a>0.5){
s=+1;
}else{
s=-1;
}
return s;

}


void Griglie(int*** Ising,int n,int N,int M){

int i,j,k;
for(i=0;i<n;i++){
for(j=0;j<N;j++){
for(k=0;k<M;k++){
Ising[i][j][k]=Spin();
}
}
}

}


void Temperature(double* T,int n){
```

```
int i;
for(i=0;i<n;i++)T[i]=Caso()*5;

}


void Bordo(int* Plus,int* Minus,int N){

int i;
for(i=0;i<N;i++){
Plus[i]=i+1;
Minus[i]=i-1;
}
Plus[N-1]=0;
Minus[0]=N-1;

}


void Bubblesort(double* T,int n){

int i,j;
double a;
for(i=0;i<n;i++){
for(j=n-1;j>i;j--){
if(T[j-1]>T[j]){
a=T[j-1];
T[j-1]=T[j];
T[j]=a;
}
}
}

}


void Metropolis(int*** Ising,int* Plus,int* Minus,double* T,int n,int N,int M, int Steps){

int istanza,i,j,passo;
int sito1,sito2;
int dx,sx,up,down;
int somma,e1,e2,delta;
double r;
double P,y,beta;
for(istanza=0;istanza<n;istanza++){
beta=(1./T[istanza]);
```

```
for(passo=0;passo<Steps;passo++){
i=(int)(Caso()*N);
j=(int)(Caso()*M);
sito1=Ising[istanza][i][j];
sito2=(-1)*sito1;
dx=Ising[istanza][Plus[i]][j];
sx=Ising[istanza][Minus[i]][j];
up=Ising[istanza][i][Minus[j]];
down=Ising[istanza][i][Plus[j]];
somma=(dx+sx+up+down);
e1=(-1)*sito1*somma;
e2=(-1)*sito2*somma;
delta=(e2-e1);
if(delta<=0){
Ising[istanza][i][j]=sito2;
}else if(delta>0){
r=Caso();
y=(-1.)*beta*delta;
P=exp(y);
if(r<=P){
Ising[istanza][i][j]=sito2;
}else{
Ising[istanza][i][j]=sito1;
}
}
}
}

}


void MagnetizzazioneMedia(double* m,int*** Ising,int n,int N,int M){

int istanza,i,j,Somma,Spin;
Spin=N*M;
for(istanza=0;istanza<n;istanza++){
Somma=0;
for(i=0;i<N;i++){
for(j=0;j<M;j++)Somma+=Ising[istanza][i][j];
}
m[istanza]=((double)Somma/Spin);
}

}
```

**Esempio di programma utilizzato per l'addestramento della mappa secondo il metodo di addestramento alternativo**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define N 220

struct neurone{
int x;
int y;
double pesi[625];
double distanza;
int flag;
};

struct segnale{
double T;
double M;
double ising[625];
};

struct bestunitmatch{
int X;
int Y;
double Pesi[625];
};

double spin(void);
void init(struct neurone* nodo,int n);
void read(struct segnale* input,int n,FILE* fp);
void bestmatch(struct neurone* nodo,int n,struct segnale input,struct bestunitmatch* bmu);
double raggio(struct neurone nodo,struct bestunitmatch bmu);
double vicinato(struct neurone nodo,struct bestunitmatch bmu,double s);

int main(void){

int i,j,k,l,step;
double s,s0;
double test1,test2,test3;
struct neurone* nodo;
nodo=calloc(225,sizeof(struct neurone));
struct segnale* input;
input=calloc(N,sizeof(struct segnale));
```

```
struct bestunitmatch* bmu;
bmu=calloc(N,sizeof(struct bestunitmatch));
double** h;
h=calloc(N,sizeof(double*));
for(i=0;i<N;i++)h[i]=calloc(225,sizeof(double));
int seed;
FILE* fp1;
FILE* fp2;
fp1=fopen("campioni6.dat","r");
fp2=fopen("SOM.dat","w");
seed=time(0);
srand48(seed);
s0=1.0;
init(nodo,225);
read(input,N,fp1);
step=7;
s=((step*s0)+0.5);
printf("Programma di addestramento inizializzato correttamente.\n");
do{
for(i=0;i<N;i++)bestmatch(nodo,225,input[i],&bmu[i]);
for(i=0;i<N;i++){
for(j=0;j<225;j++)h[i][j]=vicinato(nodo[j],bmu[i],s);
}
for(i=0;i<625;i++){
for(j=0;j<225;j++){
test1=0;
test2=0;
for(k=0;k<N;k++){
test1+=h[k][j];
test2+=h[k][j]*input[k].ising[i];
}
test3=test2/test1;
nodo[j].pesi[i]=test3;
}
}
step--;
s=((step*s0)+0.5);
}while(step>3);
printf("Addestramento completato.\n");
//STAMPA DELLA RETE
for(i=0;i<225;i++){
fprintf(fp2,"%d %d\n", nodo[i].x,nodo[i].y);
for(j=0;j<625;j++)fprintf(fp2,"%lf ", nodo[i].pesi[j]);
fprintf(fp2,"\n\n");
}
printf("Stampa della rete su file completata.\n");
```

```
printf("Programma terminato.\n");
fclose(fp1);
fclose(fp2);
return 0;

}

double spin(void){

double s;
int i;
i=lrand48()%2;
if(i==0){
s=-1.;
}else{
s=1.;
}
return s;

}

void init(struct neurone* nodo,int n){

int i,j,k,l;
k=0;
for(i=0;i<15;i++){
for(j=0;j<15;j++){
nodo[k].x=i;
nodo[k].y=j;
for(l=0;l<625;l++)nodo[k].pesi[l]=spin();
k++;
}
}
}

void read(struct segnale* input,int n,FILE* fp){

int i,j;
for(i=0;i<n;i++){
fscanf(fp,"%lf", &input[i].T);
fscanf(fp,"%lf", &input[i].M);
for(j=0;j<625;j++)fscanf(fp,"%lf", &input[i].ising[j]);
}

}
```

```
void bestmatch(struct neurone* nodo,int n,struct segnale input,struct bestunitmatch* bmu){

double test1,test2,test3;
int i,j,k;
int lulla;
int bingo;
int bongo;
int lello;
int best;
for(i=0;i<n;i++){
test3=0;
nodo[i].distanza=0;
for(j=0;j<625;j++){
test1=input.ising[j]-nodo[i].pesi[j];
test2=test1*test1;
test3+=test2;
}
nodo[i].distanza=sqrt(test3);
}
bingo=lrand48()%n;
for(i=0;i<n;i++){
nodo[i].flag=0;
if(nodo[i].distanza<nodo[bingo].distanza){
bingo=i;
}
}
lulla=0;
for(i=0;i<n;i++){
if(nodo[i].distanza==nodo[bingo].distanza){
lulla++;
nodo[i].flag=1;
}
}
bongo=(lrand48()%lulla)+1;
lello=0;
i=0;
do{
if(nodo[i].flag==1){
lello++;
}
if(lello==bongo){
best=i;
}
i++;
}while(lello!=bongo);
(*bmu).X=nodo[best].x;
```

```
(*bmu).Y=nodo[best].y;
for(i=0;i<625;i++)(*bmu).Pesi[i]=nodo[best].pesi[i];

}

double raggio(struct neurone nodo,struct bestunitmatch bmu){

double r;
int L,i,j,k,l;
double a,a1,a2;
double b,b1,b2;
L=15;
i=bmu.X;
j=bmu.Y;
k=nodo.x;
l=nodo.y;
if(i<=k){
a1=(k-i)*(k-i);
a2=(i+15-k)*(i+15-k);
(a1<a2)?(a=a1):(a=a2);
}else{
a1=(k-i)*(k-i);
a2=(i-15-k)*(i-15-k);
(a1<a2)?(a=a1):(a=a2);
}
if(j<=l){
b1=(l-j)*(l-j);
b2=(j+15-l)*(j+15-l);
(b1<b2)?(b=b1):(b=b2);
}else{
b1=(l-j)*(l-j);
b2=(j-15-l)*(j-15-l);
(b1<b2)?(b=b1):(b=b2);
}
r=sqrt(a+b);
return r;

}

double vicinato(struct neurone nodo,struct bestunitmatch bmu,double s){

double h;
double r,r1,d,e;
r1=raggio(nodo,bmu);
r=r1*r1;
d=2*s*s;
```

```
e=(-1.)*(r/d);
h=exp(e);
return h;

}
```

**Esempio di programma utilizzato per addestrare la mappa secondo la regola di apprendimento classica**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

struct neurone{
int x;
int y;
double pesi[625];
double distanza;
int flag;
};

struct segnale{
double T;
double M;
double ising[625];
};

struct bestunitmatch{
int X;
int Y;
double Pesi[625];
};

double spin(void);
void init(struct neurone* nodo,int n);
void read(struct segnale* input,int n,FILE* fp);
void bestmatch(struct neurone* nodo,int n,struct segnale input,struct bestunitmatch* bmu);
double raggio(struct neurone nodo,struct bestunitmatch bmu);
double vicinato(struct neurone nodo,struct bestunitmatch bmu,double s);

int main(void){

int i,j,k,l;
double h,m;
double a,a0,c;
```

```
double s,s0;
struct neurone* nodo;
nodo=calloc(225,sizeof(struct neurone));
struct segnale* input;
input=calloc(6225,sizeof(struct segnale));
struct bestunitmatch bmu;
int seed;
FILE* fp1;
FILE* fp2;
fp1=fopen("campioni6.dat","r");
fp2=fopen("SOM.dat","w");
seed=time(0);
srand48(seed);
a0=0.9;
c=0.9995;
s0=7.5;
init(nodo,225);
read(input,6225,fp1);
printf("Programma di addestramento inizializzato correttamente.\n");
a=a0;
s=s0;
l=0;
do{
i=lrand48()%6225;
bestmatch(nodo,225,input[i],&bmu);
for(j=0;j<225;j++){
h=vicinato(nodo[j],bmu,s);
m=a*h;
for(k=0;k<625;k++)nodo[j].pesi[k]+=m*(input[i].ising[k]-nodo[j].pesi[k]);
}
a*=c;
s*=c;
l++;
}while((s>6.4)&&(a!=0));
printf("Addestramento completato, s=%lf a=%lf l=%d.\n", s,a,l);
//STAMPA DELLA RETE
for(i=0;i<225;i++){
fprintf(fp2,"%d %d\n", nodo[i].x,nodo[i].y);
for(j=0;j<625;j++)fprintf(fp2,"%lf ", nodo[i].pesi[j]);
fprintf(fp2,"\n\n");
}
printf("Stampa della rete su file completata.\n");
printf("Programma terminato.\n");
fclose(fp1);
fclose(fp2);
return 0;
```

```
}

double spin(void){

double s;
int i;
i=lrand48()%2;
if(i==0){
s=-1.;
}else{
s=1.;
}
return s;

}

void init(struct neurone* nodo,int n){

int i,j,k,l;
k=0;
for(i=0;i<15;i++){
for(j=0;j<15;j++){
nodo[k].x=i;
nodo[k].y=j;
for(l=0;l<625;l++)nodo[k].pesi[l]=spin();
k++;
}
}
}

void read(struct segnale* input,int n,FILE* fp){

int i,j;
for(i=0;i<n;i++){
fscanf(fp,"%lf", &input[i].T);
fscanf(fp,"%lf", &input[i].M);
for(j=0;j<625;j++)fscanf(fp,"%lf", &input[i].ising[j]);
}

}

void bestmatch(struct neurone* nodo,int n,struct segnale input,struct bestunitmatch* bmu){

double test1,test2,test3;
int i,j,k;
```

```
int lulla;
int bingo;
int bongo;
int lello;
int best;
for(i=0;i<n;i++){
test3=0;
nodo[i].distanza=0;
for(j=0;j<625;j++){
test1=input.ising[j]-nodo[i].pesi[j];
test2=test1*test1;
test3+=test2;
}
nodo[i].distanza=sqrt(test3);
}
bingo=lrand48()%n;
for(i=0;i<n;i++){
nodo[i].flag=0;
if(nodo[i].distanza<nodo[bingo].distanza){
bingo=i;
}
}
lulla=0;
for(i=0;i<n;i++){
if(nodo[i].distanza==nodo[bingo].distanza){
lulla++;
nodo[i].flag=1;
}
}
bongo=(lrand48()%lulla)+1;
lello=0;
i=0;
do{
if(nodo[i].flag==1){
lello++;
}
if(lello==bongo){
best=i;
}
i++;
}while(lello!=bongo);
(*bmu).X=nodo[best].x;
(*bmu).Y=nodo[best].y;
for(i=0;i<625;i++)(*bmu).Pesi[i]=nodo[best].pesi[i];

}
```

```
double raggio(struct neurone nodo,struct bestunitmatch bmu){

double r;
int L,i,j,k,l;
double a,a1,a2;
double b,b1,b2;
L=15;
i=bmu.X;
j=bmu.Y;
k=nodo.x;
l=nodo.y;
if(i<=k){
a1=(k-i)*(k-i);
a2=(i+15-k)*(i+15-k);
(a1<a2)?(a=a1):(a=a2);
}else{
a1=(k-i)*(k-i);
a2=(i-15-k)*(i-15-k);
(a1<a2)?(a=a1):(a=a2);
}
if(j<=l){
b1=(l-j)*(l-j);
b2=(j+15-l)*(j+15-l);
(b1<b2)?(b=b1):(b=b2);
}else{
b1=(l-j)*(l-j);
b2=(j-15-l)*(j-15-l);
(b1<b2)?(b=b1):(b=b2);
}
r=sqrt(a+b);
return r;

}

double vicinato(struct neurone nodo,struct bestunitmatch bmu,double s){

double h;
double r,r1,d,e;
//PRIMA VERSIONE
/*
r1=raggio(nodo,bmu);
r=r1*r1;
d=2*s*s;
e=(-1.)*(r/d);
h=exp(e);
```

```
*/
//SECONDA VERSIONE

r1=raggio(nodo,bmu);
r=r1*r1;
d=s*s;
e=(r/d);
h=(exp((-1.)*e))*(1.-e);

return h;

}
```

**Esempio di programma utilizzato per la fase di analisi e di identificazione della transizione di fase**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include <string.h>

#define N 301

struct neurone{
int x;
int y;
double pesi[625];
double distanza;
int flag;
int linea;
};

struct segnale{
double T;
double M;
double ising[625];
};

struct bestunitmatch{
int X;
int Y;
double Pesi[625];
int Linea;
};
```

```
void init(struct neurone* nodo,int n,FILE* fp);
void read(struct segnale* input,int n,FILE* fp);
void bestmatch(struct neurone* nodo,int n,struct segnale input,struct bestunitmatch* bmu);
void temperature(double* temp,int n);

int main(void){

printf("Programma avviato.\n");

//VARIABILI NORMALI

int i,j,k,l,file;
int flagmagico[225];
int Deads;
int Alive;
double activityDeads;
double activityAlive;
double t1,t2;
t1=time(0);
t2=time(0);
printf("%lf Step VARIABILI NORMALI completato.\n", (t2-t1));

//LE SETTE RETI

struct neurone* nodo1;
nodo1=calloc(225,sizeof(struct neurone));
struct neurone* nodo2;
nodo2=calloc(225,sizeof(struct neurone));
struct neurone* nodo3;
nodo3=calloc(225,sizeof(struct neurone));
struct neurone* nodo4;
nodo4=calloc(225,sizeof(struct neurone));
struct neurone* nodo5;
nodo5=calloc(225,sizeof(struct neurone));
struct neurone* nodo6;
nodo6=calloc(225,sizeof(struct neurone));
struct neurone* nodo7;
nodo7=calloc(225,sizeof(struct neurone));
t2=time(0);
printf("%lf Step LE SETTE RETI completato.\n", (t2-t1));

//I 142 SOGGETTI DI ANALISI

struct segnale** input;
input=calloc(150,sizeof(struct segnale*));
```

```
for(i=0;i<150;i++)input[i]=calloc(5000,sizeof(struct segnale));
t2=time(0);
printf("%lf Step I 142 SOGGETTI DI ANALISI completato.\n", (t2-t1));

//VARIABILE DI APPOGGIO

struct bestunitmatch bmu;
t2=time(0);
printf("%lf Step VARIABILE DI APPOGGIO completato.\n", (t2-t1));

//LE 142 TEMPERATURE ASSOCIATE

double* temp;
temp=calloc(150,sizeof(double));
t2=time(0);
printf("%lf Step LE 142 TEMPERATURE ASSOCIATE completato.\n", (t2-t1));

//VARIABILE RANDOM

int seed;
seed=time(0);
srand48(seed);
t2=time(0);
printf("%lf Step VARIABILE RANDOM completato.\n", (t2-t1));

//CHIAMATA DELLE 7 RETI

FILE* rete1;
FILE* rete2;
FILE* rete3;
FILE* rete4;
FILE* rete5;
FILE* rete6;
FILE* rete7;
rete1=fopen("som_gradino_ciclo_unico.dat","r");
rete2=fopen("som_gradino_cinque_cicli.dat","r");
rete3=fopen("som_gauss.dat","r");
rete4=fopen("som_gauss_overtrained.dat","r");
rete5=fopen("som_polinomial_gauss.dat","r");
rete6=fopen("som_alternative_over.dat","r");
rete7=fopen("som_alternative.dat","r");
t2=time(0);
printf("%lf Step CHIAMATA DELLE 7 RETI completato.\n", (t2-t1));

//CHIAMATA DEI 142 CAMPIONI
```

```
FILE** fp;
fp=calloc(150,sizeof(FILE*));
char filename[128];
for(i=1;i<143;i++){
sprintf(filename,"casi_%d", i);
fp[i]=fopen(filename,"r");
}
t2=time(0);
printf("%lf Step CHIAMATA DEI 142 CAMPIONI completato.\n", (t2-t1));

//CHIAMATA AI 7 ANDAMENTI

FILE* analisi1;
FILE* analisi2;
FILE* analisi3;
FILE* analisi4;
FILE* analisi5;
FILE* analisi6;
FILE* analisi7;
analisi1=fopen("GRADINO1.dat","w");
analisi2=fopen("GRADINO5.dat","w");
analisi3=fopen("GAUSS.dat","w");
analisi4=fopen("OVERGAUSS.dat","w");
analisi5=fopen("POLINOGAUSS.dat","w");
analisi6=fopen("OVERMEDIUM.dat","w");
analisi7=fopen("MEDIUM.dat","w");
t2=time(0);
printf("%lf Step CHIAMATA AI 7 ANDAMENTI completato.\n", (t2-t1));

//LETTURA DELLE 142 TEMPERATURE

temperature(temp,150);
t2=time(0);
printf("%lf Step LETTURA DELLE 142 TEMPERATURE completato.\n", (t2-t1));

//LETTURA DELLE 7 RETI

init(nodo1,225,rete1);
init(nodo2,225,rete2);
init(nodo3,225,rete3);
init(nodo4,225,rete4);
init(nodo5,225,rete5);
init(nodo6,225,rete6);
init(nodo7,225,rete7);
t2=time(0);
printf("%lf Step LETTURA DELLE 7 RETI completato.\n", (t2-t1));
```

```
//LETTURA DEI 142 CAMPIONI

for(file=1;file<143;file++)read(input[file],N,fp[file]);
t2=time(0);
printf("%lf Step LETTURA DEI 142 CAMPIONI completato.\n", (t2-t1));

////////////////////////////////////////////////////////////////////////////////////////////

//ANALISI DEI DATI DA PARTE DELLA RETE 1 BASATA SUL GRADINO A CICLO UNICO

for(file=1;file<143;file++){
for(l=0;l<225;l++){
nodo1[l].flag=0;
flagmagico[l]=0;
nodo1[l].linea=l;
}
for(i=0;i<N;i++){
bestmatch(nodo1,225,input[file][i],&bmu);
j=bmu.Linea;
flagmagico[j]=1;
}
Deads=0;
for(i=0;i<225;i++){
if(flagmagico[i]==0){
Deads++;
}
}
Alive=225-Deads;
activityDeads=((double)Deads/225);
activityAlive=((double)Alive/225);
fprintf(analisi1,"%lf %d %lf %d %lf \n", temp[file],Deads,activityDeads,Alive,activityAlive)
}
t2=time(0);
printf("%lf BLOCCO 1: analisi dei dati da parte della rete 1 basata sul gradino a ciclo unic

//ANALISI DEI DATI DA PARTE DELLA RETE 2 BASATA SUL GRADINO A CINQUE CICLI

for(file=1;file<143;file++){
for(l=0;l<225;l++){
nodo2[l].flag=0;
flagmagico[l]=0;
nodo2[l].linea=l;
}
for(i=0;i<N;i++){
bestmatch(nodo2,225,input[file][i],&bmu);
```

```
j=bmu.Linea;
flagmagico[j]=1;
}
Deads=0;
for(i=0;i<225;i++){
if(flagmagico[i]==0){
Deads++;
}
}
Alive=225-Deads;
activityDeads=((double)Deads/225);
activityAlive=((double)Alive/225);
fprintf(analisi2,"%lf %d %lf %d %lf \n", temp[file],Deads,activityDeads,Alive,activityAlive)
}
t2=time(0);
printf("%lf BLOCCO 2: analisi dei dati da parte della rete 2 basata sul gradino a cinque cic

//ANALISI DEI DATI DA PARTE DELLA RETE 3 BASATA SULLA DISTRIBUZIONE GAUSSIANA

for(file=1;file<143;file++){
for(l=0;l<225;l++){
nodo3[l].flag=0;
flagmagico[l]=0;
nodo3[l].linea=l;
}
for(i=0;i<N;i++){
bestmatch(nodo3,225,input[file][i],&bmu);
j=bmu.Linea;
flagmagico[j]=1;
}
Deads=0;
for(i=0;i<225;i++){
if(flagmagico[i]==0){
Deads++;
}
}
Alive=225-Deads;
activityDeads=((double)Deads/225);
activityAlive=((double)Alive/225);
fprintf(analisi3,"%lf %d %lf %d %lf \n", temp[file],Deads,activityDeads,Alive,activityAlive)
}
t2=time(0);
printf("%lf BLOCCO 3: analisi dei dati da parte della rete 3 basata sulla distribuzione gaus

//ANALISI DEI DATI DA PARTE DELLA RETE 4 BASATA SULLA DISTRIBUZIONE GAUSSIANA SOVRALLENATA
```

```
for(file=1;file<143;file++){
for(l=0;l<225;l++){
nodo4[l].flag=0;
flagmagico[l]=0;
nodo4[l].linea=l;
}
for(i=0;i<N;i++){
bestmatch(nodo4,225,input[file][i],&bmu);
j=bmu.Linea;
flagmagico[j]=1;
}
Deads=0;
for(i=0;i<225;i++){
if(flagmagico[i]==0){
Deads++;
}
}
Alive=225-Deads;
activityDeads=((double)Deads/225);
activityAlive=((double)Alive/225);
fprintf(analisi4,"%lf %d %lf %d %lf \n", temp[file],Deads,activityDeads,Alive,activityAlive)
}
t2=time(0);
printf("%lf BLOCCO 4: analisi dei dati da parte della rete 4 basata sulla distribuzione gaus

//ANALISI DEI DATI DA PARTE DELLA RETE 5 BASATA SULLA DISTRIBUZIONE GAUSSIANA POLINOMIALE

for(file=1;file<143;file++){
for(l=0;l<225;l++){
nodo5[l].flag=0;
flagmagico[l]=0;
nodo5[l].linea=l;
}
for(i=0;i<N;i++){
bestmatch(nodo5,225,input[file][i],&bmu);
j=bmu.Linea;
flagmagico[j]=1;
}
Deads=0;
for(i=0;i<225;i++){
if(flagmagico[i]==0){
Deads++;
}
}
Alive=225-Deads;
activityDeads=((double)Deads/225);
```

```
activityAlive=((double)Alive/225);
fprintf(analisi5,"%lf %d %lf %d %lf \n", temp[file],Deads,activityDeads,Alive,activityAlive)
}
t2=time(0);
printf("%lf BLOCCO 5: analisi dei dati da parte della rete 5 basata sulla distribuzione gaus

//ANALISI DEI DATI DA PARTE DELLA RETE 6 BASATA SUL METODO DELLA MEDIA PESATA SOVRALLENATA

for(file=1;file<143;file++){
for(l=0;l<225;l++){
nodo6[l].flag=0;
flagmagico[l]=0;
nodo6[l].linea=l;
}
for(i=0;i<N;i++){
bestmatch(nodo6,225,input[file][i],&bmu);
j=bmu.Linea;
flagmagico[j]=1;
}
Deads=0;
for(i=0;i<225;i++){
if(flagmagico[i]==0){
Deads++;
}
}
Alive=225-Deads;
activityDeads=((double)Deads/225);
activityAlive=((double)Alive/225);
fprintf(analisi6,"%lf %d %lf %d %lf \n", temp[file],Deads,activityDeads,Alive,activityAlive)
}
t2=time(0);
printf("%lf BLOCCO 6: analisi dei dati da parte della rete 6 basata sul metodo della media p

//ANALISI DEI DATI DA PARTE DELLA RETE 7 BASATA SUL METODO DELLA MEDIA PESATA

for(file=1;file<143;file++){
for(l=0;l<225;l++){
nodo7[l].flag=0;
flagmagico[l]=0;
nodo7[l].linea=l;
}
for(i=0;i<N;i++){
bestmatch(nodo7,225,input[file][i],&bmu);
j=bmu.Linea;
flagmagico[j]=1;
}
```

```
Deads=0;
for(i=0;i<225;i++){
if(flagmagico[i]==0){
Deads++;
}
}
Alive=225-Deads;
activityDeads=((double)Deads/225);
activityAlive=((double)Alive/225);
fprintf(analisi7,"%lf %d %lf %d %lf \n", temp[file],Deads,activityDeads,Alive,activityAlive)
}
t2=time(0);
printf("%lf BLOCCO 7: analisi dei dati da parte della rete 7 basata sul metodo della media p

////////////////////////////////////////////////////////////////////////////////////////////

fclose(rete1);
fclose(rete2);
fclose(rete3);
fclose(rete4);
fclose(rete5);
fclose(rete6);
fclose(rete7);
printf("Step 1 completato.\n");
for(i=1;i<143;i++)fclose(fp[i]);
printf("Step 2 completato.\n");
free(fp);
printf("Step 3 completato.\n");
free(nodo1);
free(nodo2);
free(nodo3);
free(nodo4);
free(nodo5);
free(nodo6);
free(nodo7);
printf("Step 4 completato.\n");
for(i=0;i<150;i++)free(input[i]);
printf("Step 5 completato.\n");
free(input);
printf("Step 6 completato.\n");
free(temp);
printf("Step 7 completato.\n");
fclose(analisi1);
fclose(analisi2);
fclose(analisi3);
fclose(analisi4);
```

```
fclose(analisi5);
fclose(analisi6);
fclose(analisi7);
printf("Step 8 completato.\n");
t2=time(0);
printf("%lf Programma terminato.\n", (t2-t1));
return 0;

}


void init(struct neurone* nodo,int n,FILE* fp){

int i,l;
for(i=0;i<n;i++){
fscanf(fp,"%d", &nodo[i].x);
fscanf(fp,"%d", &nodo[i].y);
for(l=0;l<625;l++)fscanf(fp,"%lf", &nodo[i].pesi[l]);
}

}

void read(struct segnale* input,int n,FILE* fp){

int i,j;
for(i=0;i<n;i++){
fscanf(fp,"%lf", &input[i].T);
fscanf(fp,"%lf", &input[i].M);
for(j=0;j<625;j++)fscanf(fp,"%lf", &input[i].ising[j]);
}

}

void bestmatch(struct neurone* nodo,int n,struct segnale input,struct bestunitmatch* bmu){

double test1,test2,test3;
int i,j,k;
int lulla;
int bingo;
int bongo;
int lello;
int best;
for(i=0;i<n;i++){
test3=0;
nodo[i].distanza=0;
for(j=0;j<625;j++){
```

```
test1=input.ising[j]-nodo[i].pesi[j];
test2=test1*test1;
test3+=test2;
}
nodo[i].distanza=sqrt(test3);
}
bingo=lrand48()%n;
for(i=0;i<n;i++){
nodo[i].flag=0;
if(nodo[i].distanza<nodo[bingo].distanza){
bingo=i;
}
}
lulla=0;
for(i=0;i<n;i++){
if(nodo[i].distanza==nodo[bingo].distanza){
lulla++;
nodo[i].flag=1;
}
}
bongo=(lrand48()%lulla)+1;
lello=0;
i=0;
do{
if(nodo[i].flag==1){
lello++;
}
if(lello==bongo){
best=i;
}
i++;
}while(lello!=bongo);
(*bmu).X=nodo[best].x;
(*bmu).Y=nodo[best].y;
(*bmu).Linea=nodo[best].linea;
for(i=0;i<625;i++)(*bmu).Pesi[i]=nodo[best].pesi[i];

}

void temperature(double* temp,int n){

int i;
for(i=0;i<n;i++)temp[i]=0;
temp[1]=0.050;
temp[2]=0.100;
temp[3]=0.150;
```

```
temp[4]=0.200;
temp[5]=0.250;
temp[6]=0.300;
temp[7]=0.350;
temp[8]=0.400;
temp[9]=0.450;
temp[10]=0.500;
temp[11]=0.550;
temp[12]=0.600;
temp[13]=0.650;
temp[14]=0.700;
temp[15]=0.750;
temp[16]=0.800;
temp[17]=0.850;
temp[18]=0.900;
temp[19]=0.950;
temp[20]=1.000;
temp[21]=1.050;
temp[22]=1.100;
temp[23]=1.150;
temp[24]=1.200;
temp[25]=1.250;
temp[26]=1.300;
temp[27]=1.350;
temp[28]=1.400;
temp[29]=1.450;
temp[30]=1.500;
temp[31]=1.550;
temp[32]=1.600;
temp[33]=1.650;
temp[34]=1.700;
temp[35]=1.750;
temp[36]=1.800;
temp[37]=1.850;
temp[38]=1.900;
temp[39]=1.950;
temp[40]=2.000;
temp[41]=2.050;
temp[42]=2.100;
temp[43]=2.150;
temp[44]=2.200;
temp[45]=2.210;
temp[46]=2.220;
temp[47]=2.230;
temp[48]=2.240;
temp[49]=2.250;
```

```
temp[50]=2.260;
temp[51]=2.268;
temp[52]=2.269;
temp[53]=2.270;
temp[54]=2.271;
temp[55]=2.280;
temp[56]=2.290;
temp[57]=2.300;
temp[58]=2.310;
temp[59]=2.350;
temp[60]=2.400;
temp[61]=2.450;
temp[62]=2.500;
temp[63]=2.550;
temp[64]=2.600;
temp[65]=2.650;
temp[66]=2.700;
temp[67]=2.750;
temp[68]=2.800;
temp[69]=2.850;
temp[70]=2.900;
temp[71]=2.950;
temp[72]=3.000;
temp[73]=3.050;
temp[74]=3.100;
temp[75]=3.150;
temp[76]=3.200;
temp[77]=3.250;
temp[78]=3.300;
temp[79]=3.350;
temp[80]=3.400;
temp[81]=3.450;
temp[82]=3.500;
temp[83]=3.550;
temp[84]=3.600;
temp[85]=3.650;
temp[86]=3.700;
temp[87]=3.750;
temp[88]=3.800;
temp[89]=3.850;
temp[90]=3.900;
temp[91]=3.950;
temp[92]=4.000;
temp[93]=4.050;
temp[94]=4.100;
temp[95]=4.150;
```

```
temp[96]=4.200;
temp[97]=4.250;
temp[98]=4.300;
temp[99]=4.350;
temp[100]=4.400;
temp[101]=4.450;
temp[102]=4.500;
temp[103]=4.550;
temp[104]=4.600;
temp[105]=4.650;
temp[106]=4.700;
temp[107]=4.750;
temp[108]=4.800;
temp[109]=4.850;
temp[110]=4.900;
temp[111]=4.950;
temp[112]=5.000;
temp[113]=5.100;
temp[114]=5.200;
temp[115]=5.300;
temp[116]=5.400;
temp[117]=5.500;
temp[118]=5.600;
temp[119]=5.700;
temp[120]=5.800;
temp[121]=5.900;
temp[122]=6.000;
temp[123]=6.100;
temp[124]=6.200;
temp[125]=6.300;
temp[126]=6.400;
temp[127]=6.500;
temp[128]=6.600;
temp[129]=6.700;
temp[130]=6.800;
temp[131]=6.900;
temp[132]=7.000;
temp[133]=7.100;
temp[134]=7.200;
temp[135]=7.300;
temp[136]=7.400;
temp[137]=7.500;
temp[138]=7.600;
temp[139]=7.700;
temp[140]=7.800;
temp[141]=7.900;
```

```
temp[142]=8.000;

}
```