Studente: Manoni Leonardo
Matricola: 261049

Corso di Programmazione Procedurale e Logica
Progetto per la sessione invernale 2014/2015

DOCENTE: Prof. Marco Bernardo

SPECIFICA DEL PROBLEMA

Specifica del problema:

Scrivere una libreria ANSI C che gestisce le relazioni binarie esportando le seguenti funzioni. La prima funzione C restituisce una relazione binaria acquisita da tastiera. La seconda funzione C ha come parametro di ingresso una relazione binaria e la stampa a video. La terza funzione C ha come parametro di ingresso una relazione binaria e stabilisce se essa è una relazione d'ordine parziale, stampando a video quale proprietà non vale nel caso la relazione non sia tale. La quarta funzione C ha come parametro di ingresso una relazione binaria e stabilisce se essa è una relazione d'ordine totale, stampando a video quale proprietà non vale nel caso la relazione non sia tale. La quinta funzione C ha come parametro di ingresso una relazione binaria e stabilisce se essa è una relazione d'equivalenza, stampando a video quale proprietà non vale nel caso la relazione non sia tale. La sesta funzione C ha come parametro di ingresso una relazione binaria e stabilisce se essa è una funzione matematica; se non lo è, allora si stamperà a video quale elemento violi la proprietà, altrimenti si stamperà a video un messaggio che indica se la funzione è iniettiva, suriettiva o biiettiva.

ANALISI DEL PROBLEMA

Analisi del problema:

Analisi degli input

L'input del progetto è una relazione binaria, cioè dati due insiemi A e B, una relazione tra questi insiemi è un sottoinsieme del prodotto cartesiano A x B. Quindi gli input sono coppie di elementi.
Esempio: R={(a,b),(c,d)....}

Analisi degli output

Gli output sono costituiti dai risultati delle funzioni della libreria.

- Output Funzione numero 2: è la stampa a video della relazione binaria precedentemente acquisita nella funzione numero 1.
- Output Funzione numero 3: è la comunicazione all'utente che la relazione binaria acquisita è, o non è, una relazione d'ordine parziale, stampando a video le proprietà che non valgono nel caso la relazione non sia tale.
- Output Funzione numero 4: è la comunicazione all'utente che la relazione binaria acquisita è, o non è, una relazione d'ordine totale, stampando a video le proprietà che non valgono nel caso la relazione non sia tale.
- Output Funzione numero 5: è la comunicazione all'utente che la relazione binaria acquisita è, o non è, una relazione d'equivalenza, stampando a video le proprietà che non valgono nel caso la relazione non sia tale.
- Output Funzione numero 6: è la comunicazione all'utente che la relazione binaria acquisita è, o non è, una funzione matematica, stampando a video le coppie della relazione che non soddisfano tali proprietà, altrimenti si stamperà a video un messaggio che indica se la funzione è iniettiva, suriettiva o biiettiva.

- ***Relazioni intercorrenti tra input e output***

- L'input e l'output della funzione numero 2 sono in relazione, in quanto i dati acquisiti non sono minimamente modificati, ma stampati a video.

- La relazione tra input e output si basa sul soddisfacimento delle proprietà delle relazioni binarie, tali proprietà, data una $R \subseteq AxA$ sono:

  -RIFLESSIVA: sse $(a,a) \in R$, $\forall a \in$ campo(R).

  -SIMMETRICA: sse $(a1,a2) \in R$ implica $(a2,a1) \in R$, $\forall a1,a2 \in$ campo(R).

  -ANTISIMMETRICA: sse $(a1,a2) \in R$ implica $(a2,a1) \notin R$, $\forall a1,a2 \in$ campo(R), $a1 != a2$

  -TRANSITIVA: sse $(a1,a2) \in R$ e $(a2,a3) \in R$ implicano $(a1,a3) \in R$, $\forall a1,a2,a3 \in$ campo(R).

  -DICOTOMIA: sse $(a1,a2) \in R$ o $(a2,a1) \in R$, $\forall a1,a2 \in$ campo(R).

- La funzione numero 6, si basa sul soddisfacimento delle seguente proprietà:

  - R è una funzione da A a B, sse $\forall a \in dom(R)$ $\exists$ esattamente un $b \in B$, tale che $(a,b) \in R$.
    E in tal caso diciamo che è:

  -INIETTIVA sse $\forall b \in B$ esiste al più un $a \in dom(R)$ tale che $f(a)=b$.

  -SURIETTIVA sse $\forall b \in B$ esiste almeno un $a \in dom(R)$ tale che $f(a)=b$.

  -BIIETTIVA sse $\forall b \in B$ esiste esattamente un $a \in dom(R)$ tale che $f(a)=b$.

PROGETTAZIONE DELL'ALGORITMO

Progettazione dell'algoritmo: Principali scelte di progetto

La principale scelta di progetto è quella di operare solo con numeri interi e decimali, positivi e negativi. La struttura dati utilizzata per immagazzinare la relazione binaria è l'array bidimensionale, che ha tot righe quanto il numero di coppie e solo 2 colonne. La scelta dell'array a discapito delle liste è stata dettata dalla semplicità di confrontare due o più array potendo arrivare ad un determinato dato senza percorrere la lista intera.
Si è cercato inoltre di utilizzare meno strutture dati possibili (cercando di usare solo array e variabili) per facilitare la comprensione di altri sviluppatori che andranno a leggere il codice sorgente.

Prima Funzione: Acquisizione

-Acquisizione numero delle coppie della relazione

Come prima cosa, l'utente inserisce il numero di coppie che formerà la relazione binaria. Questa scelta è stata attuata per due motivi, il primo per facilitare l'acquisizione delle coppie di numeri, il secondo per avere la dimensione dell'array che ospiterà la relazione binaria. Il numero di coppie non può essere negativo e non deve contenere caratteri.

-Acquisizione delle coppie

L'acquisizione della relazione quindi avviene coppia per coppia, senza il bisogno di inserire parentesi, virgole o altri segni di punteggiatura, ma spaziando il primo elemento con il secondo.
Il primo elemento prenderà la posizione [x][0] dell'array bidimensionale, mentre il secondo la posizione [x][1].
Le coppie inoltre non possono essere formate da caratteri o da stringhe.

Seconda funzione: Stampa a video della relazione

-Stampa a video della relazione

Con un ciclo setacciamo l'array bidimensionale che contiene la relazione e la stampiamo a video.

Creazione campo

Il campo della relazione è l'insieme formato dall'unione del Dominio e del Codominio di R. La sua creazione facilita l'implementazione di algoritmi che serviranno più avanti per calcolare le proprietà delle relazioni binarie. Considerando che tutti gli elementi delle relazioni comprendano il Dominio e il Codominio della relazione, iniziamo mettendo tutti i primi elementi delle coppie in un array chiamato campo facendo attenzione a non creare doppioni, cioè elementi duplicati. Allo stesso modo, seguendo le opportune restrizioni mettiamo tutti i secondi elementi delle coppie in un altro array campo-b. A questo punto uniamo i due array, riscrivendo l'array campo-b in coda all'array campo.
Nota: l'array campo non può avere elementi uguali e non deve necessariamente essere ordinato in ordine crescente o decrescente.

Terza funzione: RELAZIONE D'ORDINE PARZIALE

La relazione binaria per essere una relazione d'ordine parziale deve soddisfare le seguenti proprietà: riflessiva, antisimmetrica e transitiva.

-Verifica della Proprietà Riflessiva

Attraverso due cicli confrontiamo ogni elemento del campo con tutte le coppie della relazione binaria. Se l'elemento del campo preso in esame corrisponde con una coppia della relazione, cioè prendiamo x, se tra le coppie è presente una coppia del genere (x,x), è soddisfatta la proprietà riflessiva riguardante l'elemento. A questo punto un contatore si aggiorna di un unità. Alla fine dei cicli se il contatore è uguale al numero degli elementi del campo, abbiamo la proprietà riflessiva verificata per la relazione binaria.

-Verifica della Proprietà Antisimmetrica

Con cicli annidati setacciamo l'array bidimensionale delle coppie, confrontando tutte le coppie di elementi. Nel momento in cui troviamo due coppie simmetriche, cioè (x,y) e (y,x), un contatore si aggiorna.
Finita la ricerca, se il contatore è uguale a 0, abbiamo la proprietà antisimmetrica soddisfatta.

-Verifica della Proprietà Riflessiva

Usiamo due cicli annidati per setacciare l'array delle coppie della relazione. Prendiamo una coppia, se il secondo elemento della coppia è uguale al primo elemento di un altra coppia, un contatore si aggiorna.
Una volta controllate tutte le coppie, abbiamo un contatore che sta a indicare quante coppie servono per verificare la proprietà transitiva. In poche parole andiamo a cercare, per ogni coppia (x,y), un'altra coppia che abbia come primo elemento y, cioè (y,z). Una volta trovata, il contatore si aggiorna. A questo punto ri-setacciamo l'array delle coppie per cercare la coppia che soddisfa la proprietà cioè quella coppia che ha come primo elemento il primo elemento della prima coppia (x) e come secondo elemento il secondo elemento della seconda coppia (z) --> (x,z).
Se trovato, un altro contatore si aggiorna.
Se i due contatori risultano uguali e maggiori di 0, abbiamo la proprietà transitiva.

-Verifica delle tre proprietà e stampa a video del risultato

Se le tre proprietà sono soddisfatte, stampiamo a video un messaggio che ci informa che la relazione binaria è una relazione d'ordine parziale, altrimenti stampiamo a video le proprietà che non sono soddisfatte.

Quarta funzione: RELAZIONE D'ORDINE TOTALE

La relazione binaria per essere una relazione d'ordine totale deve soddisfare le seguenti proprietà: riflessiva, antisimmetrica, transitiva e dicotomica.

-Verifica della Proprietà Riflessiva
(Come nella funzione precedente).

-Verifica della Proprietà Antisimmetrica
(Come nella funzione precedente).

-Verifica della Proprietà Transitiva
(Come nella funzione precedente)

-Verifica della Proprietà Dicotomica

Per verificare questa proprietà usiamo il campo, prendendo tutte le combinazioni di coppie possibili di elementi. Con tre cicli, due che setacciano l'array campo e uno che analizza l'array delle coppie della relazione, andiamo a verificare se i due elementi presi dal campo sono in relazione, cioè se esiste una coppia formata da entrambi gli elementi. Se esiste, un contatore si aggiorna uscendo dall'ultimo ciclo in modo da non rischiare che il contatore si aggiorni di nuovo in caso di coppia simmetrica.
A questo punto prendiamo il numero degli elementi del campo e facciamo un calcolo per sapere il numero minimo di coppie necessarie per far si che tutti gli elementi del campo siano in relazione tra loro.
Se abbiamo x elementi, il calcolo è (x-1)+(x-2)+(x-3)... finché (x-i)=0.
Esempio:
Elementi del campo = 5,
Coppie minime necessarie per essere dicotomica 4+3+2+1+0 = 10.
Se il numero delle coppie necessarie è uguale o minore al contatore precedentemente descritto, la relazione binaria soddisfa la proprietà transitiva.

-Verifica delle quattro proprietà e stampa a video del risultato

Se le quattro proprietà sono soddisfatte, stampiamo a video un messaggio che ci informa che la relazione binaria è una relazione d'ordine totale, altrimenti stampiamo a video le proprietà che non sono soddisfatte.

Quinta funzione: RELAZIONE D'EQUIVALENZA

-Verifica della Proprietà Riflessiva
(Come nella funzione precedente).
-Verifica della Proprietà Simmetrica

Con un ciclo analizziamo l'array della coppie della relazione e se ogni coppia presa in esame non risulta riflessiva, cioè se (a,b) con a != b, un contatore si aggiorna. Il passo successivo è quello di controllare se esistono coppie opposte, simmetriche alla coppia presa in considerazione cioè (b,a). Se esistono, un altro contatore si aggiorna. Se i due contatori coincidono, la proprietà simmetrica è verificata.

-Verifica della Proprietà Transitiva
(Come nella funzione precedente)

-Verifica delle tre proprietà e stampa a video del risultato

Se le tre proprietà sono soddisfatte, stampiamo a video un messaggio che ci informa che la relazione binaria è una relazione d'ordine parziale, altrimenti stampiamo a video le proprietà che non sono soddisfatte.

Sesta funzione: FUNZIONE MATEMATICA

-Verifica della proprietà per capire se la relazione è una funzione matematica

Creiamo un array di appoggio bidimensionale con 3 colonne in cui mettiamo le nostre coppie di elementi nelle prime due colonne e mettiamo momentaneamente uno 0 nell'ultima colonna. Utilizziamo l'ultima colonna come un flag di controllo. Ora confrontiamo tutte le coppie della relazione e nel momento in cui incontriamo due coppie

che hanno come primo elemento, un elemento uguale, inseriamo un 1 nella "casella di flag", cioè nella terza colonna di ciascuna riga che contiene la coppia, aumentando un contatore di un unità.
Nel momento in cui tutte le coppie sono state confrontate tra loro, se il contatore risulta = 0, la relazione binaria è una funzione matematica.

-Stampa a video del risultato

Se non è una funzione matematica, stampiamo a video le coppie di elementi che non soddisfano la proprietà descritta in precedenza.
Con un ciclo stampiamo a video tutte le coppie che hanno come flag di controllo il numero 1.

Nel caso che la relazione binaria sia una funzione matematica andiamo a verificare se si tratta di una funzione Surietiva, Iniettiva o Biettiva.

-Inseriamo il secondo insieme, l'insieme B

Acquisiamo da tastiera il numero di elementi che compone l'insieme B. Dopo aver validato il dato, acquisiamo gli elementi dell'insieme, anch'essi validati. Una seconda validazione viene fatta all'acquisizione degli elementi in quanto tutti i secondi elementi delle coppie devono far parte dell'insieme B.
il numero di elementi dell'insieme B, non può essere <=0, e non può essere un carattere. Gli elementi dell'insieme B non possono essere caratteri e devono essere inseriti in sequenza, e spaziati.

-Verifica Iniettività

Confrontiamo i secondi elementi delle coppie della relazione tra loro, se non esistono secondi elementi uguali, è verificata l'iniettività. E' sufficiente verificare solo questa proprietà, perché abbiamo già verificato in precedenza i primi elementi delle coppie quando dovevamo vedere se la relazione binaria era o meno una funzione matematica.
-Verifica Suriettività

Confrontiamo, attraverso due cicli, ogni elemento dell'insieme B con

tutti i secondi elementi delle coppie. Nel momento in cui troviamo un un uguaglianza, un contatore si aggiorna di un unità. Usciamo dal ciclo interno e proseguiamo confrontando un altro elemento con le coppie della relazione. Finiti entrambi i cicli, se il numero del contatore coincide con il numero degli elementi dell'insieme B, abbiamo suriettività.

-Verifica Biiettività

Se c'è iniettività e suriettività, abbiamo biiettività.

IMPLEMENTAZIONE DELL'ALGORITMO

Implementazione dell'algoritmo:

Di seguito viene riportato il Makefile che prevede la compilazione del file sorgente principale e della libreria secondo lo standard ANSI con segnalazione di warning:

```
progetto:  progetto.o funzioni.o Makefile
	gcc -ansi -Wall -O progetto.o funzioni.o -o progetto

progetto.o: progetto.c libreria.h Makefile
	gcc -ansi -Wall -O -c progetto.c

funzioni.o: funzioni.c libreria.h Makefile
	gcc -ansi -Wall -O -c funzioni.c

pulisci:
	rm -f progetto.o funzioni.o

pulisci_tutto:
	rm -f progetto progetto.o funzioni.o
```

Di seguito viene riportato il file libreria.h

```
extern int numero_coppie_rel(void);

extern void acquisisci_relazione(double array[][2], int numero_coppie);

extern void stampa_relazione(double array[][2], int numero_coppie);

extern void relazione_di_ordine_parziale(double array[][2],double campo[],  int
numero_coppie,int contatore[]);

extern void relazione_di_equivalenza(double array[][2],double campo[],int
numero_coppie,int contatore[]);

extern void funzione(double array[][2],int numero_coppie);

extern void relazione_di_ordine_totale(double array[][2],double campo[],int
numero_coppie,int contatore[]);

extern void crea_campo(double array[][2],double campo[],int numero_coppie,
int contatore[]);
```

Di seguito viene riportato il file sorgente della libreria: funzioni.c

```
/*inclusione librerie standar*/
#include <stdio.h>
#include <stdlib.h>

/*FUNZIONE per acquisire il numero di coppie di una relazione binaria*/
int numero_coppie_rel(void)
{

  /*dichiarazione variabili locali alla funzione*/
  int n,                    /*Input: variabile per acquisire il numero coppie*/
      esito_lettura,    /*Lavoro: flag validazione*/
      elimina_warning;  /*Lavoro: variabile che mi permette di elimare warning
derivati da scanf*/

  char carattere_non_letto; /*Lavoro: variabile da cestinare in caso di errore*/

  /*Validazione stretta dell'input, il numero_coppie non può essere negativo e
non può essere un carattere*/
  do
  {
    printf("INSERIRE IL NUMERO DI COPPIE CHE FORMANO LA RELAZIONE
BINARIA:\n");

    /*acquisisco il numero di coppie da tastiera*/
    esito_lettura = scanf("%d",
                          &n);

    if (esito_lettura != 1 ||
        n <= 0)
      do
      {

        elimina_warning = scanf("%c",
                            &carattere_non_letto);

      }/*inclusione librerie standar*/
#include <stdio.h>
#include <stdlib.h>

/*FUNZIONE per acquisire il numero di coppie di una relazione binaria*/
```

```
int numero_coppie_rel(void)
{

  /*dichiarazione variabili locali alla funzione*/
  int n,                    /*Input: variabile per acquisire il numero coppie*/
      esito_lettura,    /*Lavoro: flag validazione*/
      elimina_warning;  /*Lavoro: variabile che mi permette di elimare warning
derivati da scanf*/

  char carattere_non_letto; /*Lavoro: variabile da cestinare in caso di errore*/

  /*Validazione stretta dell'input, il numero_coppie non può essere negativo e
non può essere un carattere*/
  do
  {
    printf("INSERIRE IL NUMERO DI COPPIE CHE FORMANO LA RELAZIONE
BINARIA:\n");

    /*acquisisco il numero di coppie da tastiera*/
    esito_lettura = scanf("%d",
                          &n);

    if (esito_lettura != 1 ||
        n <= 0)
      do
      {

        elimina_warning = scanf("%c",
                            &carattere_non_letto);

      }
      while(carattere_non_letto != '\n');
  }
  while(esito_lettura != 1 ||
        n <= 0);

  elimina_warning++;
  return(n);
}
```

```
/*FUNZIONE NUMERO 1, acquisisce una relazione da tastiera e la
immagazzina in un array bidimensionale*/
void acquisisci_relazione(double array[][2], int numero_coppie)
{

  /*dichiarazione variabili locali alla funzione*/
  int i,              /*Lavoro: variabile contatore*/
      esito_lettura,   /*Lavoro: flag validazione   */
      elimina_warning; /*Lavoro: variabile che mi permette di elimare warning
derivati da scanf*/

  char carattere_non_letto;  /*Lavoro: variabile da cestinare in caso di errore*/

  printf("\nSCRIVI LE COPPIE FORMATE DA NUMERI INTERI E/O DECIMALI,
NEGATIVI E/O POSITIVI\n");

  /*acquisizione delle coppie della relazione*/
  for(i = 0;
      i < numero_coppie;
      i++)
  {
    do
    {
      printf("\nCOPPIA NUMERO %d inserisci lo spazio tra a e b:  ",
            i+1);

      /*Le coppie dei numeri vengono messe in un array bidimensionale e
vengono validate*/
      /*acquisizione del primo numero della coppia*/
      esito_lettura = scanf("%lf",
                        &array[i][0]);
      /*acquisizione del secondo numero della coppia*/
      esito_lettura = scanf("%lf",
                        &array[i][1]);

      /*abbiamo un array bidimensionale con tante righe quante sono il numero
di coppie
       e con solo 2 colonne, alla colonna 0 avremo i primi numeri delle coppie,
mentre
       alla colonna 1 avremo i secondi numeri delle coppie*/

      if (esito_lettura != 1)
```

```
        do
          elimina_warning = scanf("%c",
                                  &carattere_non_letto);
        while(carattere_non_letto != '\n');

     }
     while(esito_lettura != 1);
   }

   elimina_warning++;

   return;
}


      while(carattere_non_letto != '\n');
   }
   while(esito_lettura != 1 ||
         n <= 0);

   elimina_warning++;
   return(n);
}



/*FUNZIONE NUMERO 1, acquisisce una relazione da tastiera e la
immagazzina in un array bidimensionale*/
void acquisisci_relazione(double array[][2], int numero_coppie)
{

   /*dichiarazione variabili locali alla funzione*/
   int i,              /*Lavoro: variabile contatore*/
      esito_lettura,   /*Lavoro: flag validazione   */
      elimina_warning; /*Lavoro: variabile che mi permette di elimare warning
derivati da scanf*/

   char carattere_non_letto;  /*Lavoro: variabile da cestinare in caso di errore*/

   printf("\nSCRIVI LE COPPIE FORMATE DA NUMERI INTERI E/O DECIMALI,
NEGATIVI E/O POSITIVI\n");
```

```
  /*acquisizione delle coppie della relazione*/
  for(i = 0;
      i < numero_coppie;
      i++)
  {
    do
    {
      printf("\nCOPPIA NUMERO %d inserisci lo spazio tra a e b:  ",
            i+1);

      /*Le coppie dei numeri vengono messe in un array bidimensionale e
vengono validate*/
      /*acquisizione del primo numero della coppia*/
      esito_lettura = scanf("%lf",
                        &array[i][0]);
      /*acquisizione del secondo numero della coppia*/
      esito_lettura = scanf("%lf",
                        &array[i][1]);

      /*abbiamo un array bidimensionale con tante righe quante sono il numero
di coppie
       e con solo 2 colonne, alla colonna 0 avremo i primi numeri delle coppie,
mentre
       alla colonna 1 avremo i secondi numeri delle coppie*/

      if (esito_lettura != 1)
        do
          elimina_warning = scanf("%c",
                            &carattere_non_letto);
        while(carattere_non_letto != '\n');

    }
    while(esito_lettura != 1);
  }

  elimina_warning++;

  return;
}
```

```
/*FUNZIONE NUMERO 2, stampa a video la relazione binaria acquisita come
parametro*/
void stampa_relazione(double array[][2], int numero_coppie)
{

  /*dichiarazione variabili locali alla funzione*/
  int i;    /*Lavoro: variabile contatore*/

  /*Stampiamo a video la Relazione Binaria precedentemente acquisita*/
  printf("\nRelazione Binaria = {");

  for(i = 0;
      i < numero_coppie;
      i++)
  {

  /*stampa a video degli elementi*/
  printf("(%.2f ",
         array[i][0]);

  printf(",");

  printf(" %.2f)",
         array[i][1]);
  }

  printf("}\n\n");


  return;
}



/*Funzione per creare il campo della Relazione Binaria*/
void crea_campo(double array[][2],double campo[],int numero_coppie,int
contatore[])
{

  /*dichiarazione variabili locali alla funzione*/
  int i,                    /*Lavoro: variabile contatore*/
      j,                  /*Lavoro: variabile contatore*/
```

```
    h,                  /*Lavoro: variabile contatore*/
    v,                  /*Lavoro: variabile contatore*/
    w,                  /*Lavoro: variabile contatore*/
    cont;               /*Lavoro: variabile contatore*/

  double campo_b[numero_coppie];    /*Lavoro: array di appoggio*/

  /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
  v = 0;

  /*Creiamo il campo*/
  /*mettiamo tutti gli elementi in posizione 0 dell'array bidimensionale in un
secondo array monodimensionale
  controllando che non ci siano doppioni, l'array dovrà contenere gli elementi
del campo*/
  for(i = 0;
      i < numero_coppie;
      i++)
  {

    h = 0;
    for (j = 0;
         j <= numero_coppie;
         j++)
    {
      if (campo[j] == array[i][0])
        h = h + 1;
    }

    if (h == 0)  /*controllo elementi, se un elemento uguale è già stato inserito,
non reinserire l'elemento nell'array*/
    {
      campo[v] = array[i][0];
      v++;     /*all'uscita dei for v è il numero di posizione in cui è contenuto
l'ultimo dato dell'array*/
    }
  }

  /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
  w = 0;

  /*Inseriamo in un array di appoggio gli elementi in posizione 1 dell'array
```

```
bidimensionale*/
  for(i = 0;
      i < numero_coppie;
      i++)
  {
    h = 0;
     for (j = 0 ;
          j <= numero_coppie;
          j++)
     {
       if (campo_b[j] == array[i][1])
         h = h + 1;
     }

     if (h == 0)  /*anche in questo caso effettuiamo un controllo con gli elementi
già inseriti*/
     {          /*per non creare doppioni*/
       campo_b[w] = array[i][1];
       w++;
     }
  }

  /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
  cont = 0;

  /*mettiamo gli elementi dell'array di appoggio campo_b in coda all'array
campo sempre controllando
    che non ci siamo doppioni, quindi facciamo un controllo incrociato tra i due
array utilizzando la
    variabile contatore v come ultima posizione dell'array campo*/
  for(i = 0;
      i < w;
      i++)
  {
    h = 0;
    for (j = 0;
         j < v;
         j++)
    {
      if (campo_b[i] == campo[j])
        h = h + 1;
    }
```

```
    if (h == 0)
    {
      campo[v+cont] = campo_b[i]; /*v+cont è la prima posizione disponibile in
campo*/
      cont = cont+1;

    }
  }

  contatore[0] = v+cont;

  return;
}



/*FUNZIONE NUMERO 3, stabilisce se la relazione binaria è una relazione
d'ordine Parziale*/
/*In tal caso deve soddisfare le seguenti proprietà: RIFLESSIVA,
ANTISIMMETRICA e TRANSITIVA*/
void relazione_di_ordine_parziale(double array[][2], double campo[], int
numero_coppie,int contatore[])
{

  /*dichiarazione variabili locali alla funzione*/
  int i,                    /*Lavoro: variabile contatore*/
      h,                    /*Lavoro: variabile contatore*/
      cont_riflessiva,      /*Lavoro: variabile contatore*/
      cont_antisimmetrica,  /*Lavoro: variabile contatore*/
      cont_transitiva,      /*Lavoro: variabile contatore*/
      j,                   /*Lavoro: variabile contatore*/
      r;                    /*Lavoro: variabile contatore*/


  /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
  cont_riflessiva = 0;

  /*controlliamo se ci sono coppie con elementi uguali per vedere se è
soddisfatta la
    proprietà RIFLESSIVA (a,a) */
```

```
for(i = 0;
    i < contatore[0];
    i++)
{
  for (j = 0;
       j < numero_coppie;
       j++)
  {
    if ((array[j][0] == campo[i]) &&
        (array[j][1] == campo[i]))

      cont_riflessiva = cont_riflessiva + 1;
  }
}


/*settiamo a 0 la variabile contatore per evitare errori di core dump*/
cont_antisimmetrica = 0;

/*Controlliamo se la proprietà ANTISIMMETRICA sia soddisfatta*/
for(i = 0;
    i < numero_coppie;
    i++)
{
  for (j = i;
       j<numero_coppie;
       j++)
  {
    if ((array[i][0] == array[j+1][1]) &&
        (array[i][1] == array[j+1][0]))

      cont_antisimmetrica=cont_antisimmetrica+1;
  }
}

/*settiamo a 0 la variabile contatore per evitare errori di core dump*/
cont_transitiva = 0;
r = 0;

/*Controlliamo se la proprietà TRANSITIVA sia soddisfatta*/
for(i = 0;
    i < numero_coppie;
```

```
     i++)
  {
    if(array[i][0] != array[i][1])
    {
      for (j = 0;
           j < numero_coppie;
           j++)
      {
        if(array[i][1] == array[j][0] &&
           array[j][1] != array[j][0] &&
           array[j][1] != array[i][0])
       {
          r = r + 1;

          for(h = 0;
              h < numero_coppie;
              h++)
          {
            if (array[i][0] == array[h][0] &&
              array[j][1] == array[h][1])
            {
              cont_transitiva = cont_transitiva + 1;
            }
          }
       }
     }
   }
  }

  /*se il numero di coppie è minore di 3, non può esistere la proprietà
transitiva*/
  if (numero_coppie < 3)
    r = 0;

  /*se le seguenti proprietà sono state riscontrate, abbiamo una relazione
d'ordine parziale*/
  if ((cont_transitiva == r)&&
      (r > 0)&&
      (cont_riflessiva  == contatore[0])&&
      (cont_antisimmetrica == 0))

    printf("\n La relazione Binaria è una RELAZIONE D'ORDINE PARZIALE \n");
```

```
  else

    printf("\nNON è una RELAZIONE D'ORDINE PARZIALE\n Non soddisfa le
seguenti proprietà: \n");

  /*stampa a video delle proprietà non soddisfatte dalla Relazione Binaria */
  if (cont_riflessiva  != contatore[0])
    printf("\n- RIFLESSIVA\n");

  if (cont_antisimmetrica != 0)
    printf("\n- ANTISIMMETRICA\n");

  if ((cont_transitiva != r) ||
      (r == 0 ))
    printf("\n- TRANSITIVA\n");

  printf("\n\n");

  return;
 }

/*FUONZIONE NUMERO 4, stabilisce se la relazione Binaria è una Relazione
d'ordine TOTALE*/
/*In tal caso deve soddisfare le seguenti proprietà: RIFLESSIVA,
ANTISIMMETRICA, TRANSITIVA E DICOTOMIA*/
void relazione_di_ordine_totale(double array[][2],double campo[], int
numero_coppie,int contatore[]){

  /*dichiarazione variabili locali alla funzione*/
  int i,                    /*Lavoro: variabile contatore*/
      j,                   /*Lavoro: variabile contatore*/
      h,                    /*Lavoro: variabile contatore*/
      c,                    /*Lavoro: variabile contatore*/
      k,                   /*Lavoro: variabile di appoggio*/
      r,                   /*Lavoro: variabile contatore*/
      cont_riflessiva,      /*Lavoro: variabile contatore*/
      cont_antisimmetrica,  /*Lavoro: variabile contatore*/
      cont_transitiva,      /*Lavoro: variabile contatore*/
      cont_dicotomia,       /*Lavoro: variabile contatore*/
      totale_relazioni;     /*Lavoro: variabile di appoggio*/
```

```
  /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
  cont_riflessiva = 0;

  /*controlliamo se ci sono coppie con elementi uguali per vedere se è
soddisfatta la
    proprietà RIFLESSIVA (a,a) */
  for(i = 0;
      i < contatore[0];
      i++)
  {
    for (j = 0;
         j < numero_coppie;
         j++)
    {
      if (array[j][0] == campo[i] &&
          array[j][1] == campo[i])
          cont_riflessiva = cont_riflessiva + 1;
    }
  }


  /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
  cont_antisimmetrica = 0;

  /*Controlliamo se la proprietà ANTISIMMETRICA sia soddisfatta*/
  for(i = 0;
      i < numero_coppie;
      i++)
  {
    for (j = i;
         j < numero_coppie;
         j++)
    {
      if (array[i][0] == array[j+1][1] &&
          array[i][1] == array[j+1][0])
            cont_antisimmetrica = cont_antisimmetrica + 1;
    }/*Se la proprietà non è stata riscontrata, il contatore si aggiorna di 1*/
  }


  /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
```

```
cont_transitiva = 0;
r = 0;

/*Controlliamo se la proprietà TRANSITIVA sia soddisfatta*/
for(i = 0;
    i < numero_coppie;
    i++)
{
  if(array[i][0] != array[i][1])
  {
    for (j = 0;
         j < numero_coppie;
         j++)
    {
      if(array[i][1] == array[j][0] &&
         array[j][1] != array[j][0] &&
         array[j][1] != array[i][0])
      {
        r = r + 1;

        for(h = 0;
            h < numero_coppie;
            h++)
        {
          if (array[i][0] == array[h][0] &&
              array[j][1] == array[h][1])
          {
            cont_transitiva = cont_transitiva + 1;

          }
        }
      }
    }
   }
}

/*Se il numero delle coppie è >3, non può esistere la proprietà transitiva*/
if (numero_coppie < 3)
 r = 0;

/*Controlliamo che la relazione soffisfi la proprietà di dicotomia*/
```

```
  /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
  cont_dicotomia = 0;

  /*ora possiamo verificare la proprietà di DICOTOMIA in quanto possiamo
verificare se tutti gli elementi del campo sono in relazione*/
  /*con tre for annidati scorriamo l'array campo prendendo due numeri alla
volta, che andranno confrontati
   con la relazione. se tutte le combinazioni tra due numeri del campo
corrispondono con almeno una coppia
   di elementi della relazione, abbiamo la proprietà di DICOTOMIA*/
  for(i = 0;
      i < contatore[0];
      i++)
  {
    for (j = i + 1;
         j < contatore[0];
         j++)
    {
      for (c = 0;
           c < numero_coppie;
           c++)
      {
        if ((array[c][0] == campo[i] && /*controllo incrociato*/
             array[c][1] == campo[j]) ||
            (array[c][0] == campo[j] &&
             array[c][1] == campo[i]))
        {
          cont_dicotomia = cont_dicotomia + 1;   /*la variabile contatore si
aggiorna di uno nel momento in cui c'è un uguaglianza*/
          c = numero_coppie;                  /*il contatore del for viene settato in
modo da uscire */
        }

      }
    }
  }


  /*calcoliamo il numero di uguaglianze che servono per soddisfare la
proprietà
  se abbiamo 3 elementi, avremo 2+1+0=3 uguaglianze, 4 elementi 3+2+1+0=6
uguaglianze*/
```

```
  totale_relazioni = 0;
  i = 0;
  k = contatore[0] - 1;

  for(i = 0;
      i < contatore[0];
      i++)
  {
    totale_relazioni = totale_relazioni + (k-i);
  }

  /*se le seguenti proprietà sono state riscontrate, abbiamo una relazione
d'ordine parziale*/
  if ((cont_transitiva == r) &&
      (r > 0) &&
      (cont_riflessiva == contatore[0]) &&
      (cont_antisimmetrica == 0) &&
      (cont_dicotomia >= totale_relazioni))

    printf("\nLa relazione Binaria è una RELAZIONE D'ORDINE TOTALE\n");

  else

    printf("\nNON è una RELAZIONE D'ORDINE TOTALE\n Non soddisfa le
seguenti proprietà: \n");


  /*stampa a video delle proprietà non soddisfatte dalla Relazione Binaria */
  if (cont_riflessiva != contatore[0])
    printf("\n- RIFLESSIVA\n");

  if (cont_antisimmetrica != 0)
    printf("\n- ANTISIMMETRICA\n");

  if ((cont_transitiva != r) ||
      (r == 0))
     printf("\n- TRANSITIVA\n");

  if (cont_dicotomia < totale_relazioni)
    printf("\n- DICOTOMIA\n");

  printf("\n\n");
```

```
  return;
}



/*FUNZIONE NUMERO 5, stabilisce se la relazione binaria è una relazione di
equivalenza*/
/*in tal caso deve soddisfare le seguenti proprietà: RIFLESSIVA,
ANTISIMMETRICA, TRANSITIVA*/
void relazione_di_equivalenza(double array[][2],double campo[],int
numero_coppie,int contatore[])
{

/*dichiarazione variabili locali alla funzione*/
  int i,                    /*Lavoro: variabile contatore*/
      h,                    /*Lavoro: variabile contatore*/
      cont_riflessiva,      /*Lavoro: variabile contatore*/
      cont_simmetrica,      /*Lavoro: variabile contatore*/
      cont_transitiva,      /*Lavoro: variabile contatore*/
      j,                    /*Lavoro: variabile contatore*/
      r,                    /*Lavoro: variabile contatore*/
      m;                    /*Lavoro: variabile contatore*/

  /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
  cont_riflessiva = 0;

  /*controlliamo se ci sono coppie con elementi uguali per vedere se è
soddisfatta la
  proprietà RIFLESSIVA (a,a) */
  for(i = 0;
      i < contatore[0];
      i++)
  {
    for (j = 0;
         j < numero_coppie;
         j++)
    {
      if (array[j][0] == campo[i] &&
          array[j][1] == campo[i])
          cont_riflessiva = cont_riflessiva + 1;
```

```
  }
}

/*settiamo a 0 la variabile contatore per evitare errori di core dump*/
cont_simmetrica = 0;
m = 0;

/*Controlliamo se la proprietà SIMMETRICA sia soddisfatta*/
for(i = 0;
    i < numero_coppie;
    i++)
{
  if(array[i][0] != array[i][1])
  {
    m = m + 1;
    for (j = 0;
         j < numero_coppie;
         j++)
    {
      if ((array[i][0] == array[j][1]) &&
          (array[i][1] == array[j][0]))

            cont_simmetrica = cont_simmetrica + 1;
    }
  }
}

/*settiamo a 0 la variabile contatore per evitare errori di core dump*/
cont_transitiva = 0;
r = 0;
/*Controlliamo se la proprietà TRANSITIVA sia soddisfatta*/
for(i = 0;
    i < numero_coppie;
    i++)
{
  if(array[i][0] != array[i][1])
  {
    for (j = 0;
         j < numero_coppie;
         j++)
    {
```

```
      if(array[i][1] == array[j][0] &&
         array[j][1] != array[j][0] &&
         array[j][1] != array[i][0])
      {
        r = r + 1;

        for(h = 0;
            h < numero_coppie;
            h++)
        {
          if (array[i][0] == array[h][0] &&
              array[j][1] == array[h][1])
          {
            cont_transitiva = cont_transitiva + 1;
          }
        }
      }
    }
  }
 }

  /*se il numero di coppie è minore di 3, non può esistere la proprietà
transitiva*/
  if (numero_coppie < 3)
    r = 0;

  /*se le seguenti proprietà sono state riscontrate, abbiamo una relazione
d'ordine parziale*/
  if ((cont_transitiva == r)&&
      (r > 0)&&
      (cont_riflessiva == contatore[0])&&
      (cont_simmetrica == m))

    printf("\nLa relazione Binaria è una RELAZIONE D'EQUIVALENZA\n");

  else

    printf("\nNON è una RELAZIONE D'EQUIVALENZA\n Non soddisfa le
seguenti proprietà: \n");

  /*stampa a video delle proprietà non soddisfatte dalla Relazione Binaria */
  if (cont_riflessiva != contatore[0])
```

```
    printf("\n- RIFLESSIVA\n");

  if (cont_simmetrica != m )
    printf("\n- SIMMETRICA\n");

  if ((cont_transitiva != r) ||
      (r == 0 ))
    printf("\n- TRANSITIVA\n");

  printf("\n\n");

  return;
}

/*FUNZIONE NUMERO 6, stabilisce se la relazione binaria è una FUNZIONE
MATEMATICA */
void funzione(double array[][2],int numero_coppie)
{

  /*dichiarazione delle variabili locali alla funzione*/
  int i,    /*Lavoro: variabile contatore*/
      j,    /*Lavoro: variabile contatore*/
      h,    /*Lavoro: variabile contatore*/
      numero_elementi_in_b,    /*Input: numero di elementi che formano
l'insieme B*/
      cont_insieme_b,         /*Lavoro: variabile contatore*/
      cont_iniettiva,         /*Lavoro: variabile contatore*/
      cont_suriettiva,        /*Lavoro: variabile contatore*/
      esito_lettura_0,        /*Lavoro: flag validazione*/
      elimina_warning;        /*Lavoro: variabile che mi permette di elimare
warning derivati da scanf*/


  double support[numero_coppie][3];   /*Lavoro: array multidimensionale di
appoggio con 3 colonne e
                                righe uguali al numero delle coppie della relazione
binaria */

  char carattere_non_letto_0,   /*Lavoro: flag validazione*/
       carattere_non_letto;     /*Lavoro: flag validazione*/

  /*settiamo il contatore a 0 per evitare errori di core dump*/
```

```
  h = 0;
  j = 0;

  /*controlliamo se la relazione è una Funzione matematica*/
  for(i = 0;
      i<numero_coppie;
      i++)
  {
    support[i][0] = array[i][0];
    support[i][1] = array[i][1];

  /*settiamo un flag=0 di fianco ad ogni coppia di elementi*/
  support[i][2] = 0;
  }

  for(i = 0;
      i<numero_coppie;
      i++)
  {
    for(j = i + 1;
        j < numero_coppie;
        j++)
    {
      if (array[i][0] == array[j][0])
      {
        /*se l'elemento in posizione 0 è uguale a un altro elemento della stessa
posizione di una coppia diversa,
          mettiamo flag=1 aggiornando un contatore h*/
        support[i][2] = 1;
        support[j][2] = 1;
        h = h + 1;
      }
    }
  }

  /*se il contatore è diverso da 0, la relazione binaria non è una funzione*/
  if (h != 0)
  {
    printf("\nLa Relazione Binaria non è una FUNZIONE MATEMATICA!\nLe
coppie di elementi che violano la proprietà sono:\n");
    for(i = 0;
        i<numero_coppie;
```

```
       i++)
   {
    /*andiamo a stampare a video tutte le coppie che non soddisfano la
proprietà, quelle coppie con flag=1*/
    if (support[i][2] == 1)
    {
     printf("-(%.2f\t%.2f)\n",
           support[i][0],
           support[i][1]);
    }
   }

   printf("\n");
  }
  else
  {

   printf("\nLa relazione Binaria è una FUNZIONE MATEMATICA \n");

   printf("\nINSIEME B");
   /*Inserire in numero di elementi presenti nell'insieme B ed eventuale
validazione*/
   printf("\nInserire il numero di elementi che formano l'insieme B\n");
   do
   {
    esito_lettura_0 = scanf("%d",
                    &numero_elementi_in_b);

    if(esito_lettura_0 != 1 ||
      numero_elementi_in_b <= 0)
     do
      elimina_warning = scanf("%c",
                      &carattere_non_letto_0);
     while(carattere_non_letto_0 != '\n');
   }
   while(esito_lettura_0 != 1 ||
        numero_elementi_in_b <= 0);
   /*il numero degli elementi appena acquisito non può essere un carattere,
non può essere uguale e minore di zero*/

  /*dichiarazione array statico per immagazzinare gli elementi dell'insieme B*/
  double insieme_b[numero_elementi_in_b];
```

```
  printf("\n\nInserire gli elementi in sequenza separati da uno spazio e
premere invio \n");
  /*acquisizione elementi e validazione input*/
  do
  {
    for(i = 0;
        i<numero_elementi_in_b;
        i++)
    {

      esito_lettura_0 = scanf("%lf",
                          &insieme_b[i]);
    }

      if (esito_lettura_0 != 1)
        do

          elimina_warning =  scanf("%c",
                             &carattere_non_letto);

        while(carattere_non_letto != '\n');
  }
  while(esito_lettura_0 != 1);
  /*gli elementi acquisiti non possono essere caratteri*/

  /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
  cont_insieme_b=0;

  /*Validazione elementi insieme B, se i secondi elementi delle coppie inserite,
non fanno parte dell'insieme B, abbiamo un errore*/
  for(i = 0;
      i < numero_elementi_in_b;
      i++)
  {
    for(j = 0;
        j < numero_coppie;
        j++)
    {

      /*confrontiamo gli elementi inseriti nell'array dell'insieme B con i dati in
seconda colonna dell'array delle coppie*/
```

```
    if (insieme_b[i] == array[j][1])
    {
      /*ogni elemento che corrisponde aggiorna il contatore di un unità*/
      cont_insieme_b++;
    }

  }
 }

 if (cont_insieme_b != numero_coppie)
 {

   printf("\n\nErrore di inserimento dell'insieme B. Tutti i secondi elementi
delle coppie devono essere inseriti nell'insieme B.");
   printf("\nPremere 6 e riscrivere l'insieme B");

 }
 else
 {
   /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
   cont_iniettiva=0;

   /*verifichiamo l'iniettività*/
   for(i = 0;
       i < numero_coppie;
       i++)
       {
         for(j = i+1;
             j < numero_coppie;
             j++)
         {
           /*confrontiamo tutti i secondi elementi delle coppie, se ne esistono
due uguali, non c'è iniettività*/
           if (array[i][1] == array[j][1])
             cont_iniettiva++;
         }
       }


   /*settiamo a 0 la variabile contatore per evitare errori di core dump*/
   cont_suriettiva = 0;
```

```
    /*verifichiamo la suriettività*/
    for(i = 0;
        i < numero_elementi_in_b;
        i++)
    {
      for(j = 0;
          j < numero_coppie;
          j++)
      {

        /*confrontiamo gli elementi dell'insieme B con i secondi elementi delle
coppie*/
        if (insieme_b[i] == array[j][1])
        {
          cont_suriettiva++;
          /*se esistono elementi confrontati uguali, aggiorniamo il contatore e
usciamo dal contatore*/
          j = numero_coppie;
        }

      }
    }

    /*se la funzione è iniettiva e suriettiva, abbiamo biiettività*/
    if(cont_suriettiva == numero_elementi_in_b &&
       cont_iniettiva == 0)
    {
      printf("La funzione Matematica è:\n-BIIETTIVA");
    }
    else
    {
       if (cont_suriettiva==numero_elementi_in_b || cont_iniettiva==0 )
       {
         printf("\nLa funzione Matematica è:\n");

           if (cont_suriettiva == numero_elementi_in_b)

             printf("\n-SURIETTIVA\n");

           if(cont_iniettiva==0)

             printf("\n-INIETTIVA\n");
```

```
        }
        else
        {

          printf("\nLa funzione Matematica non è iniettiva e non è suriettiva.\n");

        }
      }

   }
  }

  printf("\n\n");

  elimina_warning++;
return;
}
```

Di seguito viene riportato il file sorgente: progetto.c

```
/*PROGETTO PPL SESSIONE INVERNALE 2014/2015*/

/****************************************/
/****** Autore: Manoni Leonardo*********/
/****************************************/



/**************************/
/***inclusione librerie****/
/**************************/

#include <stdio.h>
#include <stdlib.h>
#include "libreria.h"


/* Dichiarazione delle funzioni */
int numero_coppie_rel(void);

void acquisisci_relazione(double array[][2], int numero_coppie);

void stampa_relazione(double array[][2], int numero_coppie);

void relazione_di_ordine_parziale(double array[][2],double campo[],int
numero_coppie, int contatore[]);

void relazione_di_ordine_totale(double array[][2],double campo[], int
numero_coppie,int contatore[]);

void relazione_di_equivalenza(double array[][2],double campo[], int
numero_coppie,int contatore[]);

void funzione(double array[][2],int numero_coppie);

void scrivi_legenda();

void crea_campo(double array[][2],double campo[],int numero_coppie,int
contatore[]);
```

```
/*********************************/
/*********Funzione main*********/
/*********************************/

int main(void){

  /* Dichiarazione delle variabili locali alla funzione */
  int numero_coppie,contatore[1],   /*Input: struttura dati che immagazzina le coppie di elementi*/
    elimina_warning;              /*Lavoro: variabile di lavoro per eliminare i warning*/

  char scelta;    /*Lavoro: variabile di appoggio, per scegliere l'istruzione*/

  printf("\nProgramma di testing per il corretto funzionamento della libreria\n");
  printf("\n\nProgetto PPL sessione invernale 2014/2015\n-Autore: Manoni Leonardo\n\n\n");

  /*chiamata funzione per acquisire il numero di coppie della relazione binaria*/
  numero_coppie = numero_coppie_rel();

  /*dichiarazione array bidimensionale che avrà tante righe quanto il numero di coppie della relazioni e solo due colonne*/
  double array[numero_coppie][2];
  double campo[numero_coppie*2];

  /*FUNZIONE NUMERO 1, acquisisce in un array bidimensionale la relazione binaria*/
  acquisisci_relazione(array,numero_coppie);

  /*funzione che crea il campo*/
  crea_campo(array,campo,numero_coppie,contatore);

  scrivi_legenda();

  do
  {
    elimina_warning = scanf("%c",&scelta);

    if (scelta=='2')
```

```
   {

     /*FUNZIONE NUMERO 2, stampa a video la relazione binaria acquisita
come parametro*/
     stampa_relazione(array, numero_coppie);

     scrivi_legenda();



   }

   if (scelta=='3')
   {

     /*FUNZIONE NUMERO 3, stabilisce se la relazione binaria è una relazione
d'ordine Parziale*/
     relazione_di_ordine_parziale(array,campo,numero_coppie,contatore);

     scrivi_legenda();
   }

   if (scelta=='4')
   {

     /*FUNZIONE NUMERO 4, stabilisce se la relazione binaria è una relazione
d'ordine Totale*/
     relazione_di_ordine_totale(array,campo,numero_coppie,contatore);

     scrivi_legenda();

   }

   if (scelta=='5')
   {

     /*FUNZIONE NUMERO 5, stabilisce se la relazione binaria è una relazione
d'Equivalenza*/
     relazione_di_equivalenza(array,campo,numero_coppie,contatore);

     scrivi_legenda();
```

```
    }

    if (scelta=='6')
    {

      /*FUNZIONE NUMERO 6, stabilisce se la relazione binaria è una Funzione
Matematica*/
      funzione(array,numero_coppie);

      scrivi_legenda();
    }


  }
  while(scelta != '0');
  /*con la digitazione dello 0, si esce dal ciclo, quindi dal programma*/

  /*istruzione non significativa, ma utile per eliminare i warning*/
  elimina_warning++;

  return(0);
}


/*funzione che richiama la legenda*/
void scrivi_legenda(){

  printf("Selezionare:\n");
  printf("\n 2- Per stampare la RELAZIONE BINARIA acquisita");
  printf("\n 3- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'ORDINE
PARZIALE");
  printf("\n 4- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'ORDINE
TOTALE");
  printf("\n 5- Per stabilire se RELAZIONE BINARIA è una RELAZIONE
D'EQUIVALENZA");
  printf("\n 6- Per stabilire se RELAZIONE BINARIA è una FUNZIONE
MATEMATICA");
  printf("\n\n 0- Per uscire dal programma\n\n");

return;
}
```

TESTING DEL PROGRAMMA

Testing del programma:

Vengono di seguito mostrati alcuni test significativi a dimostrazione del corretto funzionamento del programma.

Compilazione tramite makefile secondo lo standard ANSI con segnalazione di warning.

```
leo@leo-HP-650-Notebook-PC:~/Scrivania/programmazione$ make
gcc -ansi -Wall -O -c progetto.c
gcc -ansi -Wall -O -c funzioni.c
gcc -ansi -Wall -O progetto.o funzioni.o -o progetto
leo@leo-HP-650-Notebook-PC:~/Scrivania/programmazione$ make
```

Lancio dell'eseguibile e schermata iniziale.

```
leo@leo-HP-650-Notebook-PC:~/Scrivania/programmazione$ ./progetto

Programma di testing per il corretto funzionamento della libreria


Progetto PPL sessione invernale 2014/2015
-Autore: Manoni Leonardo
```

Validazione del numero delle coppie inserito.

```
Progetto PPL sessione invernale 2014/2015
-Autore: Manoni Leonardo


INSERIRE IL NUMERO DI COPPIE CHE FORMANO LA RELAZIONE BINARIA:
0
INSERIRE IL NUMERO DI COPPIE CHE FORMANO LA RELAZIONE BINARIA:
-1
INSERIRE IL NUMERO DI COPPIE CHE FORMANO LA RELAZIONE BINARIA:
a
INSERIRE IL NUMERO DI COPPIE CHE FORMANO LA RELAZIONE BINARIA:
\
INSERIRE IL NUMERO DI COPPIE CHE FORMANO LA RELAZIONE BINARIA:
ppap
INSERIRE IL NUMERO DI COPPIE CHE FORMANO LA RELAZIONE BINARIA:
?
INSERIRE IL NUMERO DI COPPIE CHE FORMANO LA RELAZIONE BINARIA:
3

SCRIVI LE COPPIE FORMATE DA NUMERI INTERI E/O DECIMALI, NEGATIVI E/O POSITIVI
```

Validazione acquisizione coppie della relazione binaria.

```
SCRIVI LE COPPIE FORMATE DA NUMERI INTERI E/O DECIMALI, NEGATIVI E/O POSITIVI

COPPIA NUMERO 1 inserisci lo spazio tra a e b:  e 6

COPPIA NUMERO 1 inserisci lo spazio tra a e b:  9 t

COPPIA NUMERO 1 inserisci lo spazio tra a e b:  \ é

COPPIA NUMERO 1 inserisci lo spazio tra a e b:  1,2

COPPIA NUMERO 1 inserisci lo spazio tra a e b:  1 2

COPPIA NUMERO 2 inserisci lo spazio tra a e b:
```

Esempio di input corretto.

```
INSERIRE IL NUMERO DI COPPIE CHE FORMANO LA RELAZIONE BINARIA:
5

SCRIVI LE COPPIE FORMATE DA NUMERI INTERI E/O DECIMALI, NEGATIVI E/O POSITIVI

COPPIA NUMERO 1 inserisci lo spazio tra a e b:  1 3

COPPIA NUMERO 2 inserisci lo spazio tra a e b:  2 4

COPPIA NUMERO 3 inserisci lo spazio tra a e b:  2 2

COPPIA NUMERO 4 inserisci lo spazio tra a e b:  3 4

COPPIA NUMERO 5 inserisci lo spazio tra a e b:  1 4
```

Stampa a video della relazione acquisita. (Funzione numero 2).

```
Selezionare:

 2- Per stampare la RELAZIONE BINARIA acquisita
 3- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'ORDINE PARZIALE
 4- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'ORDINE TOTALE
 5- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'EQUIVALENZA
 6- Per stabilire se RELAZIONE BINARIA è una FUNZIONE MATEMATICA

 0- Per uscire dal programma

2

Relazione Binaria = {(1.00 , 3.00)(2.00 , 4.00)(2.00 , 2.00)(3.00 , 4.00)(1.00 , 4.00)}
```

Effettuiamo qualche test per ogni funzione.

TEST 1 funzione numero 3 (Relazione d'ordine Parziale)
Numero Coppie: 5
Relazione: {(1,1),(1,2),(2,3),(1,3)(3,1)}

```
NON è una RELAZIONE D'ORDINE PARZIALE
 Non soddisfa le seguenti proprietà:

- RIFLESSIVA

- ANTISIMMETRICA

- TRANSITIVA
```

Non è una relazione d'ordine parziale, non soddisfa nessuna delle tre proprietà in quanto è riflessiva solo per l'elemento 1, manca la transitività tra le coppie (2,3) e (3,1) nonostante ci sia per le coppie (1,2) e (2,3). Le coppie (1,3) e (3,1) sono simmetriche.

TEST 2 funzione numero 3 (Relazione d'ordine Parziale)
Numero Coppie: 5
Relazione: {(1,1),(1,2),(2,3),(3,3)(2,2)}

```
NON è una RELAZIONE D'ORDINE PARZIALE
 Non soddisfa le seguenti proprietà:

- TRANSITIVA
```

Non è una relazione d'ordine parziale, non soddisfa la proprietà transitiva. Le coppie (1,2) (2,3) non sono transitive in quanto tra la relazione binaria manca la coppia (1,3).

TEST 3 funzione numero 3 (Relazione d'ordine Parziale)
Numero Coppie: 6
Relazione: {(1,1),(1,2),(2,3),(3,3)(2,2),(1,3)}

```
Relazione Binaria = {(1.00 , 1.00)(1.00 , 2.00)(2.00 , 3.00)(1.00 , 3.00)(2.00 , 2.00)(3.00 , 3.00)}

Selezionare:

 2- Per stampare la RELAZIONE BINARIA acquisita
 3- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'ORDINE PARZIALE
 4- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'ORDINE TOTALE
 5- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'EQUIVALENZA
 6- Per stabilire se RELAZIONE BINARIA è una FUNZIONE MATEMATICA

 0- Per uscire dal programma

3

 La relazione Binaria è una RELAZIONE D'ORDINE PARZIALE
```

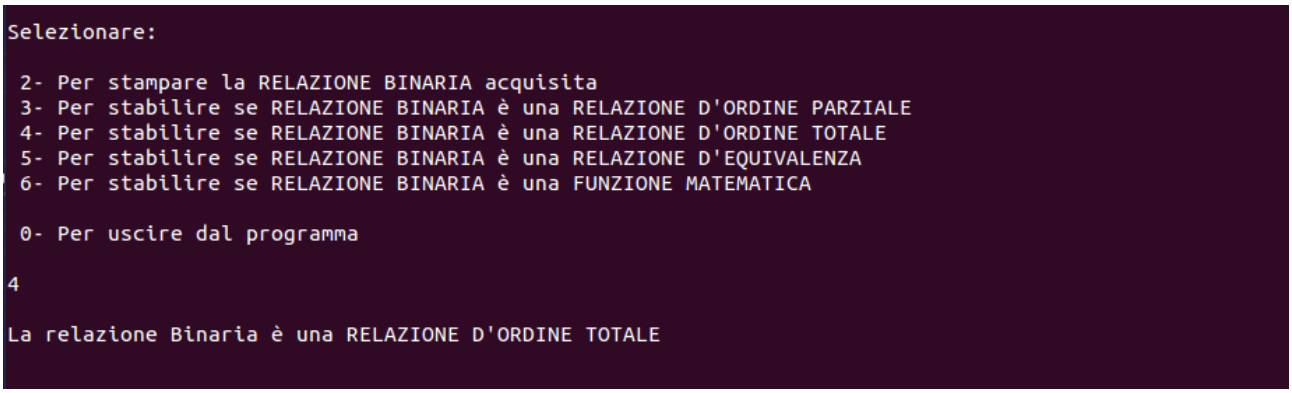
TEST 1 funzione numero 4 (Relazione d'ordine Totale)
Numero Coppie: 5
Relazione: {(1,1),(3,1),(4,4),(2,2),(1,3)}

Non soddifa nessuna proprietà in quanto non è riflessiva rispetto l'elemento 3, le coppie (1,3)(3,1) sono simmetriche, non esiste transitività, è non c'è dicotomia in quanto tutti gli elementi del campo, non sono in relazione tra di loro.

TEST 2 funzione numero 4 (Relazione d'ordine Totale)
Numero Coppie: 7
Relazione: {(1,1),(3,3),(4,4),(2,2),(3,4)(2,4),(2,3)}

Non è verificata la proprietà di dicotomia in quanto l'elemento 1, non è in relazione con nessuno degli altri elementi.

TEST 3 funzione numero 4 (Relazione d'ordine Totale)
Numero Coppie: 6
Relazione: {(1,1),(1,2),(2,3),(3,3)(2,2),(1,3)}



TEST 1 funzione numero 5 (Relazione d'Equivalenza)
Numero Coppie: 6
Relazione: {(1,1),(1,2),(2,3),(3,3)(2,2),(1,3)}

```
NON è una RELAZIONE D'EQUIVALENZA
 Non soddisfa le seguenti proprietà:

- SIMMETRICA
```

Non è una relazione d'equivalenza, non soddisfa la proprietà simmetrica in quanto le coppie (1,2),(2,3),(1,3) non sono simmetriche.
TEST 2 funzione numero 5 (Relazione d'Equivalenza)
Numero Coppie: 6
Relazione: {(2,1),(1,2),(2,3),(3,2)(3,1),(1,3)}

```
NON è una RELAZIONE D'EQUIVALENZA
 Non soddisfa le seguenti proprietà:

- RIFLESSIVA
```

Non soddisfa la proprietà riflessiva di nessuno degli elementi del campo.

TEST 3 funzione numero 5 (Relazione d'Equivalenza)
Numero Coppie: 9
Relazione: {(2,1),(1,2),(2,3),(3,2)(3,1),(1,3),(1,1),(3,3),(2,2)}

```
Relazione Binaria = {(1.00 , 2.00)(2.00 , 1.00)(2.00 , 3.00)(3.00 , 2.00)(1.00 , 3.00)(3.00 , 1.00)(1.00 , 1.00)(2.00 , 2.00)(3.00 , 3.00)}

Selezionare:

 2- Per stampare la RELAZIONE BINARIA acquisita
 3- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'ORDINE PARZIALE
 4- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'ORDINE TOTALE
 5- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'EQUIVALENZA
 6- Per stabilire se RELAZIONE BINARIA è una FUNZIONE MATEMATICA

 0- Per uscire dal programma

5

La relazione Binaria è una RELAZIONE D'EQUIVALENZA
```

TEST 1 funzione numero 6 (Funzione Matematica)
Numero Coppie: 6
Relazione: {(2,1),(1,5),(5,3),(6,2)(2,8)(4,3)}

```
Relazione Binaria = {(2.00 , 1.00)(1.00 , 5.00)(5.00 , 3.00)(6.00 , 2.00)(2.00 , 8.00)(4.00 , 3.00)}

Selezionare:

 2- Per stampare la RELAZIONE BINARIA acquisita
 3- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'ORDINE PARZIALE
 4- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'ORDINE TOTALE
 5- Per stabilire se RELAZIONE BINARIA è una RELAZIONE D'EQUIVALENZA
 6- Per stabilire se RELAZIONE BINARIA è una FUNZIONE MATEMATICA

 0- Per uscire dal programma

6

La Relazione Binaria non è una FUNZIONE MATEMATICA!
Le coppie di elementi che violano la proprietà sono:
-(2.00  1.00)
-(2.00  8.00)
```

Nel caso in cui la relazione binaria sia una funzione matematica, inseriamo l'insieme B. Validazione inserimento numero elementi dell'insieme B.

```
La relazione Binaria è una FUNZIONE MATEMATICA

INSIEME B
Inserire il numero di elementi che formano l'insieme B
r
-9
0
\
4


Inserire gli elementi in sequenza separati da uno spazio e premere invio
```

Validazione inserimento elementi dell'insieme B.

```
La relazione Binaria è una FUNZIONE MATEMATICA

INSIEME B
Inserire il numero di elementi che formano l'insieme B
3


Inserire gli elementi in sequenza separati da uno spazio e premere invio
q w r
3 e f
\ 3 \
1 2 r
2,3,4
2 3 4

La funzione Matematica è:
```

TEST 2 funzione numero 6 (Funzione Matematica)
Numero Coppie: 3
Relazione: {(2,3),(4,5),(5,4)}
Insieme B: {2,3,4,5}

```
La relazione Binaria è una FUNZIONE MATEMATICA

INSIEME B
Inserire il numero di elementi che formano l'insieme B
4


Inserire gli elementi in sequenza separati da uno spazio e premere invio
2 3 4 5

La funzione Matematica è:

-INIETTIVA
```

TEST 3 funzione numero 6 (Funzione Matematica)
Numero Coppie: 4
Relazione: {(2,3),(3,4),(4,5),(6,3)}
Insieme B: {3,4,5}

```
La relazione Binaria è una FUNZIONE MATEMATICA

INSIEME B
Inserire il numero di elementi che formano l'insieme B
3


Inserire gli elementi in sequenza separati da uno spazio e premere invio
3 4 5

La funzione Matematica è:

-SURIETTIVA
```

TEST 4 funzione numero 6 (Funzione Matematica)
Numero Coppie: 3
Relazione: {(2,3),(3,4),(4,5)}
Insieme B: {3,4,5}

```
La relazione Binaria è una FUNZIONE MATEMATICA

INSIEME B
Inserire il numero di elementi che formano l'insieme B
3


Inserire gli elementi in sequenza separati da uno spazio e premere invio
3 4 5
La funzione Matematica è:
-BIIETTIVA
```

VERIFICA DEL PROGRAMMA

Verifica del programma:

Di seguito è riportata la validazione di una parte del programma tramite la tripla di Hoare. Si dice tripla di Hoare una tripla nella seguente forma:

$$\{Q\}\ S\ \{R\}$$

Dove Q è un predicato detto precondizione, S è un'istruzione ed R è un predicato detto postcondizione. La tripla è vera se e solo se l'esecuzione dell'istruzione S inizia in uno stato della computazione in cui Q è soddisfatta, e termina raggiungendo uno stato della computazione in cui R è soddisfatta.

- Istruzione selezionata:

```
m = 0;
if (array[i][0] != array[i][1])
{
   m = m + 1;
}
```

sia array[i][0] ≠ array[i][1] V array[i][0] = array[i][1]

wp(S,R) = ((β → wp(S,R)) ∧ ((¬β) → wp(S,R)))

R => (m = 0) V (m = 1)

P => ((array[i][0] ≠ array[i][1]) → ((m = 1) V (m = 0))
((array[i][0] == array[i][1]) → ((m = 1) V (m = 0))

ovvero:

((array[i][0] ≠ array[i][1]) → ((m = 1) V (m = 0)) = Vero
((array[i][0] == array[i][1]) → ((m = 1) V (m = 0)) = Vero
(Vero ∧ Vero) = Vero