University of Urbino

applied computer science

Procedural and logic programming

Report

Project for the 2014/2015 winter session

Studente:
Marco Tamagno
matricola no: 261985

Studente:
Francesco Belacca
matricola no: 260492

Lecturer:
Marco Bernardo

January 11, 2015

Contents

1 Specifica del Problema

Write an ANSI C library that manages binary relations by exporting the following functions. The rst C function returns a binary relation introduced through the keyboard. The second C function has a binary relation as input parameter and prints it to the screen. The third C function has a binary relation as input parameter and establishes whether it is a partial order relation, printing to the screen which property does not hold in the case that the relation is not such. The fourth C function has a binary relation as input parameter and establishes whether it is a total order relation, printing to the screen which property does not hold in the case that the relation is not such. The fth C function has a binary relation as input parameter and establishes whether it is an equivalence relation, printing to the screen which property does not hold in the case that the relation is not such. The sixth C function has a binary relation as input parameter and establishes whether it is a mathematical function; if it is not, then the element violating the property will be printed to the screen, otherwise a message will be printed to the screen indicating whether the function is injective, surjective, or bijective. [The project can be submitted also by rst-year students.]

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 ‘e una relazione d′ ordine parziale, stampando a video quale propriet‘a 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 ‘e una relazione d′ ordine totale, stampando a video quale propriet‘a 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 ‘e una relazione d′ equivalenza, stampando a video quale proprieta‘ 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 ‘e una funzione matematica; se non lo ‘e, allora si stampera‘ a video quale elemento violi la propriet‘a, altrimenti si stamper‘a a video un messaggio che indica se la funzione ‘e iniettiva, suriettiva o biiettiva. [Il progetto pu‘o essere consegnato anche da studenti del primo anno.]

2 Analisi del Problema

2.1 Input

1. Per l′ acquisizione come input abbiamo una relazione binaria del tipo (a,b) che viene acquisita da tastiera;
2. Come input per le altre 5 funzioni abbiamo una relazione(precedentemente esportata dalla prima).

2.2 Output

1. La prima funzione(Acquisizione) restituisce una funzione binaria acquisita da tastiera;

2. La seconda funzione(Stampa) non restituisce nulla, ma stampa a video la relazione che aveva in ingresso;//
3. La terza funzione "ordine parziale" non restituisce nulla, ma stampa a video se la Relazione binaria acquisita è di ordine parziale o meno e stampa a video le propietà della funzione, per poter mostrare a schermo quali popietà non vengono rispettate;

4. La quarta funzione (ordine totale) non restituisce nulla, ma stampa a video se la Relazione binaria acquisita è di ordine totale o meno,stampando a video quale proprieta′ non vale nel caso la relazione non sia tale;

5. La quinta funzione(relazione equivalenza) non restituisce nulla, ma stampa a video se la relazione binaria acquisita è una relazione di equivalenza o meno e stampa a video le propietà della funzione, per poter mostrare a schermo quali popietà non vengono rispettate;

6. la sesta funzione(check funzione) non restituisce nulla, ma stampa a video se la relazione binaria acquisita è una funzione, e in caso contrario stampa a video quale coppia non fa rispettare le propietà.

3 Progettazione dell′ Algoritmo

3.1 Teoria

Per lo sviluppo di questo programma si necessita di alcuni cenni di Teoria degli insiemi quali:

Concetto di Relazione Binaria : In matematica, una relazione binaria definita su di un insieme, anche detta relazione o corrispondenza tra due oggetti, è un elenco di coppie ordinate di elementi appartenenti all'insieme. In modo equivalente, una relazione binaria è un sottoinsieme del prodotto cartesiano di un insieme con se stesso.

Concetto di Relazione d′ Ordine Parziale: In matematica, pi precisamente in teoria degli ordini, una relazione d'ordine o ordine su di un insieme è una relazione binaria tra elementi appartenenti all'insieme che gode delle seguenti proprietà:
riflessiva
antisimmetrica
transitiva.

Concetto di Relazione d′ Ordine Totale: Una relazione d′ ordine si dice Totale, quando oltre a essere parziale soddisfa anche la propietà di Dicotomia (tutti gli elementi devono essere in relazione tra di loro).

Concetto di riflessività : In logica e in matematica, una relazione binaria R in un insieme X è detta riflessiva se ogni elemento di X è in tale relazione con se stesso.

Concetto di transitività: In matematica, una relazione binaria R in un insieme X è transitiva se e solo se per ogni a, b, c appartenenti ad X, se a è in relazione con b e b è in relazione con c, allora a è in relazione con c.

Concetto di simmetricità: In matematica, una relazione binaria R in un insieme X è simmetrica se e solo se, presi due elementi qualsiasi a e b, vale che se a è in relazione con b allora anche b è in relazione con a.

Concetto di funzione: In matematica, una funzione, anche detta applicazione, mappa o trasformazione, è definita dai seguenti oggetti:
Un insieme X detto dominio della funzione. * Un insieme Y detto codominio della funzione. * Una relazione f : X -¿ Y che ad ogni elemento dell'insieme X associa uno ed un solo elemento dell'insieme Y ; l'elemento assegnato a x appartenente ad X tramite f viene abitualmente indicato con f(x) .

Concetto di Iniettività: Una funzione si dice iniettiva quando a ogni elemento del dominio è assegnato uno e uno solo elemento del codominio.

Concetto di Suriettività: Una funzione si dice suriettiva quando ogni elemento del codominio viene raggiunto da un elemento del dominio.

3.2 Funzioni per l'acquisizione:

acquisizione() : per acquisire la relazione.

3.3 Funzioni per la verifica delle proprietà:

check_iniettivita() : per controllare se l′ iniettività è rispettata o meno (0 non c′ è, 1 c′ è).

check_transitivita() : per controllare se la transitività viene rispettata o meno (0 non c′ è, 1 c′ è).

check_simmetria() : per controllare se la simmetria viene rispettata o meno (0 non c′ è, 1 c′ è).

check_riflessivita() : per controllare se la riflessività viene rispettata o meno (0 non c′ è, 1 c′ è).

check_dicotomia() : per verificare se la dicotomia viene rispettata o meno (0 non c′ è, 1 c′ è).

check_suriettivita(): verifica se la funzione gode della proprietà di suriettività, in questo caso sarà sempre settata a 1 in quanto tutti gli elementi del codominio (presi come gli elementi dei vari secondi termini digitati durante l′ acquisizione) avranno sempre un elemento del dominio associato(dato che non si può acquisire il secondo termine se non se ne acquisice prima il relativo primo, o arrivare alla funzione check_suriettivita() avendo acquisito solo il primo).

3.4 Funzioni principali:

ordine_parziale() : richiama le funzioni delle proprietà e controlla se c′ è un ordine parziale(stampa a video se c′ è o meno un ordine parziale, e nel caso non c′ è stampa quali proprietà non vengono rispettate).

ordine_totale(): richiama la funzione ordine_parziale e check_dicotomia e controlla se c′ è un ordine totale(stampa a video se esiste o meno un ordine totale, e nel caso non c′ è stampa quali propietà non vengono rispettate).

relazione_equivalenza() : richiama le funzioni delle proprietà e controlla se c′ è una relazione d′ equivalenza(stampa a video se c′ è o meno una relazione d′ equivalenza, e nel caso non c′ è stampa quali proprietà non vengono rispettate).

check_funzione():verifica se la relazione è una funzione(stampa a video se c′ è o non c′ è una funzione e nel caso non ci sia dice quale coppia non soddisfa le proprietà).

3.5 Input

Per l′ input abbiamo necessità di usare una struttura dati dinamica, nella quale andiamo a salvare la Relazione Binaria dataci dall′ utente, il numero delle coppie e il tipo di input (numerico o per stringhe).

L input dovrà essere dotato di diversi controlli, se l′ utente sceglie di inserire un input di tipo numerico allora non potra digitare stringhe e/o caratteri speciali etc.

La scelta di due tipi di input differente dovrà essere data per dare la possibilità all′ utente nel caso scelga di fare un′ input di tipo numerico di poter effettuare operazioni non legate alle funzioni della libreria, (esempio : l′ utente vuole decidere di moltiplicare l′ input per due, e vedere se mantiene le propietà, con un′ input di tipo numerico l′ utente pu farlo e ci avrebbe un senso, con un′ input di tipo stringa meno).

La scelta dell′ input di tipo stringa dovrà essere data per aver maggior completezza, una relazione binaria non deve essere forzatamente numerica ma pu essere anche tra cose, oggetti, animali, colori e qualsiasi altra cosa possa venire in mente.

Alle varie funzioni verrà data come input la struttura dati salvata in precedenza dalla funzione Acquisizione, per poterne verificare le varie propietà.

3.6 Output - Acquisizione

Durante l′ acquisizione avremo diversi output video (printf) che guideranno l′ utente nell′ inserimento dei dati, e che segnaleranno eventuali errori commessi. Finita l′ acquisizione dovremo restituire l′ indirizzo della struttura, che all′ interno quindi conterra′ i dati inseriti dall′ utente. Abbiamo scelto di fare ci perchè non essendo permesso l′ utilizzo di variabili globali, il modo pi semplice di passare i dati inseriti da una funzione all′ altra e′ quello di creare una struttura dinamica. Una volta restituito l′ indirizzo della struttura, a seconda della funzione lanciata nel file Test.c si lanceranno le altre 5 funzioni, dato che queste prendono tutte in pasto l′ output della prima (cioè l′ indirizzo della struttura della relazione binaria) e la utilizzano per verificarne varie proprieta′ .

3.7 Output - stampa

La funzione stampa avra′ come output la stampa a video della struttura acquisita, con qualche aggiunta grafica(le parentesi e le virgole) per rendere il tutto pi facilmente interpretabile e leggibile.

3.8 Output - ordine_parziale

La funzione ordine_parziale avra′ come output la stampa a video del risultato della verifica delle proprieta′ di riflessivita′ antisimmetria e transitivita′ . Nel caso in cui siano tutte verificate si stampera′ che la relazione e′ una relazione di ordine parziale, mentre nel caso in cui non siano verificate si stampera′ che non lo e′ e il perche′ (cioe′ quale proprieta′ non e′ o non sono verificate).

3.9 Output - ordine_totale

La funzione ordine_totale avra′ come output la stampa a video del risultato della verifica delle proprieta′ necessarie ad avere una relazione d′ ordine parziale, e verifichera′ poi se anche la dicotomia è valida per la relazione o meno. Nel caso in cui tutto sia positivo, allora si stampera′ che la relazione e′ di ordine totale, mentre se non lo e′ si stampera′ cosa fa in modo che non lo sia.

3.10 Output - relazione_equivalenza

La funzione relazione_equivalenza avra′ come output la stampa a video del risultato della verifica delle proprieta′ di riflessivita′ simmetria e transitivita′ e nel caso in cui siano tutte positive si stampera′ che la relazione e′ una relazione di equivalenza, mentre nel caso in cui qualcosa non sia verificato si stampera′ cio′ che impedisce alla relazione di essere una relazione d′ equivalenza.

3.11 Output - check_funzione

La funzione check_funzione avra′ come output la stampa a video della verifica della proprieta′ che rende la relazione binaria una funzione, e in caso lo sia anche se questa e′ suriettiva(che poi spiegheremo essere sempre verificata) e iniettiva, e in caso sia entrambe si stampera′ che la relazione binaria oltre ad essere una funzione e′ una funzione biiettiva.

4 Implementazione dell′ algoritmo

4.1 Libreria

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4
5  /*******************STRUTTURA relBin
       ************************/
6  /***** Creo una struttura dove salvare le coppie
       appartenenti alla Relazione*******/
7
8  struct relBin{
9    /****** Coppia Numerica ******/
10   double  *primo_termine,
11       *secondo_termine;
12
13   /***** Coppia Qualsiasi******/
14   char **prima_stringa,
15      **seconda_stringa;
16
17   /**** Variabili per salvare se ho acquisito una
         coppia numerica o no e il numero delle coppie****
         */
18   int controllo,
19     dimensione;
20 };
21
22 /*DICHIARO LE FUNZIONI*/
23 int check_simmetria(struct relBin);
24 int check_riflessivita(struct relBin);
25 int check_transitivita(struct relBin);
26 int check_suriettivita(struct relBin);
27 void check_biiettivita(struct relBin);
28
29 /******************Funzione di acquisizione
       ******************/
30
31 struct relBin acquisizione(struct relBin relazione){
32
33 int acquisizione_finita = 0;
34 int scan = 0;
35
```

```
36
37 relazione.dimensione = 0;
38 relazione.primo_termine = (double *) malloc(2);
39 relazione.secondo_termine = (double *) malloc(2);
40 relazione.prima_stringa = (char **) malloc(100);
41 relazione.seconda_stringa = (char **) malloc(100);
42
43  while((relazione.controllo < 1) || (relazione.
       controllo > 2) || scan != 1){
44 fflush(stdin);
45 printf("\n Premi 1 se vuoi immettere solo numeri, 2
     per altro\n ");
46 printf("\n scelta: ");
47 scan = scanf("%d",&relazione.controllo);
48 }
49
50 /** risetto scan a 0 **/
51 scan=0;
52
53 /*Acquisizione Numerica*/
54
55 if(relazione.controllo == 1){
56   while(acquisizione_finita == 0){
57     relazione.dimensione++;
58     acquisizione_finita = 2;
59
60  /*Acquisisco il primo termine della coppia*/
61
62     printf("\n Inserisci il primo termine della coppia
         \n ");
63     relazione.primo_termine = (double *) realloc(
         relazione.primo_termine, (relazione.dimensione
         +1) * sizeof(double));
64     scan = 0;
65  /*Check del primo termine della coppia*/
66
67   while(scan != 1){
68     printf("  Primo Termine: ");
69     fflush(stdin);
70     scan = scanf("%lf",&relazione.primo_termine[
         relazione.dimensione - 1]);
71     if(scan == 0)
72     printf("\n C'e' un errore, reinserire il primo
         termine\n");
```

```
73      }
74
75   /*Acquisisco il secondo termine della coppia*/
76      scan = 0;
77      printf("\n Inserisci il secondo termine della
            coppia \n ");
78      relazione.secondo_termine = (double *) realloc(
            relazione.secondo_termine, (relazione.
            dimensione+1) * sizeof(double));
79
80  /*Check del secondo termine della coppia*/
81
82    while(scan != 1){
83      printf("  Secondo Termine: ");
84      fflush(stdin);
85      scan = scanf("%lf",&relazione.secondo_termine[
            relazione.dimensione - 1]);
86      if(scan == 0)
87      printf("\n C'e' un errore, reinserire il secondo
            termine\n");
88      }
89
90   /*Chiedo all'utente se ci sono altre coppie*/
91
92    while(acquisizione_finita < 0 || acquisizione_finita
         > 1 || scan != 1){
93      printf("\n Vuoi acquisire un'altra coppia? immetti
            1 per uscire, 0 per continuare\n ");
94      printf("\n scelta: ");
95      fflush(stdin);
96      scan = scanf("%d",&acquisizione_finita);
97      }
98    }
99  }
100
101 /*risetto scan a 0*/
102 scan = 0;
103
104 /*Acquisizione con stringhe*/
105 if(relazione.controllo == 2){
106   while(acquisizione_finita == 0){
107     relazione.dimensione++;
108     acquisizione_finita = 2;
109
```

```
110   /*Acquisisco il primo termine della coppia*/
111
112       printf("␣Inserisci␣il␣primo␣termine␣della␣coppia␣\
              n␣");
113       printf("␣␣Primo␣Termine:␣");
114        relazione.prima_stringa[relazione.dimensione - 1]
               = (char *) malloc(50);
115       scan = scanf("␣%[^\n]s",relazione.prima_stringa[
              relazione.dimensione - 1]);
116
117   /*Acquisisco il secondo termine della coppia*/
118
119       printf("␣Inserisci␣il␣secondo␣termine␣della␣coppia
              ␣\n␣");
120       printf("␣␣Secondo␣Termine:␣");
121        relazione.seconda_stringa[relazione.dimensione -
               1] = (char *) malloc(50);
122       scan = scanf("␣%[^\n]s",relazione.seconda_stringa[
              relazione.dimensione - 1]);
123
124  /*risetto scan a 0*/
125  scan = 0;
126
127   /*Chiedo all'utente se ci sono altre coppie*/
128
129       while(acquisizione_finita < 0 ||
              acquisizione_finita > 1 || scan != 1){
130
131         printf("\n␣Vuoi␣acquisire␣un'altra␣coppia?␣
                 immetti␣1␣per␣uscire,␣0␣per␣continuare\n");
132         scan = scanf("%d",&acquisizione_finita);
133         }
134       }
135    }
136
137  printf("\n\n␣␣␣...␣Acquisizione␣Terminata␣...\n\n");
138  return relazione;
139  }
140
141  /*****************************FUNZIONE DI STAMPA
        ******************************/
142
143  void stampa(struct relBin stampa){
144
```

```
145   int i = 0;
146
147   printf("\n La relazione binaria e ':");
148   printf("\n\n  {");
149
150  /******Stampa per coppie numeriche *****/
151
152     if(stampa.controllo == 1){
153       while(i < stampa.dimensione){
154
155           printf(" (%.2lf,%.2lf)",stampa.primo_termine[i
                ],stampa.secondo_termine[i]);
156         if(i+1 != stampa.dimensione)
157             printf(" ;");
158         i++;
159      }
160     }
161
162   /********Stampa  per coppie non numeriche *********/
163
164    if(stampa.controllo == 2){
165        while(i < stampa.dimensione){
166          printf("(%s,%s)",stampa.prima_stringa[i],
                stampa.seconda_stringa[i]);
167          if(i+1 != stampa.dimensione)
168           printf(";");
169       i++;
170
171       }
172    }
173
174  /***************** Fine Stampa ******************/
175
176    printf(" }\n");
177    printf("\n\n    ... Stampa Terminata ...\n\n");
178
179  }
180
181  /*****************FUNZIONE DI VERIFICA DI RELAZIONI D
       'ORDINE******************/
182
183  int ordine_parziale(struct relBin verifica){
184
185    int riflessivita ,
```

```
186       transitivita,
187       simmetria,
188       parziale;
189
190     /*STAMPO LE PROPIETA' DELLA RELAZIONE*/
191
192     printf("\n\n La relazione:\n\n");
193
194   /********* Chiamo le funzioni per poter stabilire le
          propiet **********/
195
196     riflessivita = check_riflessivita(verifica);
197     simmetria = check_simmetria(verifica);
198     transitivita = check_transitivita(verifica);
199
200   /************ Controllo se rispetta le propiet per
          essere una relazione d'ordine parziale***********/
201
202     if(transitivita == 1 && simmetria == 0 &&
          riflessivita == 1){
203       parziale = 1;
204       printf("\n Quindi e' una relazione d'ordine
             parziale\n\n");
205     }
206     else{
207
208       printf("\n Non e' una relazione d'ordine parziale
             in quanto non rispetta tutte le propieta'\n");
209       parziale = 0;
210     }
211       if(transitivita == 0)
212       printf("\n manca la propieta' di transitivita'\n")
             ;
213       if(simmetria == 1)
214       printf("\n manca la propieta' di asimmetria\n");
215       if(riflessivita == 0)
216       printf("\n manca la propieta' di riflessivita'\n")
             ;
217   /************ Fine controllo Ordine Parziale
          *********************/
218
219     printf("\n\n    ... Controllo Ordine Parziale
          Terminato ...\n\n\n\n");
220     return(parziale);
```

```
221 }
222
223
224 /**********FUNZIONE PER CONTROLLARE LA RIFLESSIVIT
      **********/
225
226 int check_riflessivita (struct relBin verifica){
227
228   int i,
229     j,
230     k,
231     riscontro,
232     secondo_riscontro,
233     riflessivita;
234
235   riflessivita = 1;
236   i = 0;
237   j = 0;
238   k = 0;
239   riscontro = 0;
240   secondo_riscontro = 0;
241
242 /* Verifica riflessivit */
243
244 /* Definizione: una relazione per la quale esiste
      almeno un elemento che non e' in relazione con s
      stesso non soddisfa la definizione di riflessivit
      */
245
246   while((i < verifica.dimensione) && (k < verifica.
        dimensione)){
247
248 /* Verifica riflessivit per numeri*/
249
250     if(verifica.controllo == 1){
251       riscontro = 0;
252       secondo_riscontro = 0;
253       if(verifica.primo_termine[i] == verifica.
            secondo_termine[i])
254         riscontro++; /**** Controllo se c' stato un
              riscontro a,a****/
255         secondo_riscontro++;
256       if(riscontro != 0){
257         i++;
```

```
258           k++;
259         }
260       /**/
261         else{
262           j=0;
263           riscontro = 0;
264           secondo_riscontro = 0;
265
266 /***************** Controllo la riflessivit per gli
      elementi del primo insieme
      ****************************/
267
268           while(j < verifica.dimensione){
269             if(j == i)
270               j++;
271             else{
272               if(verifica.primo_termine[i] == verifica.
                    primo_termine[j])
273                 if(verifica.primo_termine[j] == verifica
                      .secondo_termine[j])
274                   riscontro++;
275
276             j++;
277           }
278         }
279
280         j = 0;
281
282 /***************** Controllo la riflessivit per gli
      elementi del secondo insieme
      ****************************/
283
284           while(j < verifica.dimensione){
285             if(j == k)
286               j++;
287             else{
288               if(verifica.secondo_termine[k] == verifica
                    .secondo_termine[j])
289                 if(verifica.primo_termine[j] == verifica
                      .secondo_termine[j])
290                   secondo_riscontro++;
291
292             j++;
293           }
```

```
294         }
295           if(riscontro != 0)
296             i++;
297
298   /**** Se non c' stato un riscontro di riflessivit
        esco e setto la riflessivit a 0 *****/
299
300           else{
301             i=verifica.dimensione;
302             riflessivita = 0;
303           }
304
305           if(secondo_riscontro != 0)
306             k++;
307
308           else{
309             k=verifica.dimensione;
310             riflessivita = 0;
311           }
312         }
313
314       }
315
316     /*************** VERIFICA RIFLESSIVIT PER STRINGHE
          *****************/
317
318     if(verifica.controllo == 2){
319       riscontro = 0;
320       secondo_riscontro = 0;
321       if(strcmp(verifica.prima_stringa[i],verifica.
            seconda_stringa[i]) == 0)
322         riscontro++;
323         secondo_riscontro++;
324       if(riscontro != 0){
325         i++;
326         k++;
327       }
328
329       else{
330         j=0;
331         riscontro = 0;
332         secondo_riscontro = 0;
333
```

```
334 /**************** Controllo la riflessivit per gli
      elementi del primo insieme
      ****************************/
335
336       while(j < verifica.dimensione){
337         if(j == i)
338           j++;
339         else{
340           if(strcmp(verifica.prima_stringa[i],verifica
                .prima_stringa[j]) == 0)
341             if(strcmp(verifica.prima_stringa[j],
                  verifica.seconda_stringa[j]) == 0)
342               riscontro++;
343
344           j++;
345         }
346       }
347
348       j = 0;
349
350 /**************** Controllo la riflessivit per gli
      elementi del secondo insieme
      ****************************/
351
352       while(j < verifica.dimensione){
353         if(j == k)
354         j++;
355         else{
356           if(strcmp(verifica.seconda_stringa[k],
                verifica.seconda_stringa[j]) == 0)
357             if(strcmp(verifica.prima_stringa[j],
                  verifica.seconda_stringa[j]) == 0)
358               secondo_riscontro++;
359
360           j++;
361         }
362       }
363       if(riscontro != 0)
364         i++;
365
366       else{
367         i=verifica.dimensione;
368         riflessivita = 0;
369       }
```

```
370
371         if(secondo_riscontro != 0)
372           k++;
373
374         else{
375           k=verifica.dimensione;
376           riflessivita = 0;
377         }
378       }
379
380     }
381
382 }
383
384 /********* Controllo se    riflessiva
      ******************/
385
386     if(riflessivita == 1)
387       printf("    e' riflessiva\n");
388     else
389       printf("    non e' riflessiva\n");
390
391 /********** Fine riflessivita **********************
      */
392
393     return(riflessivita);
394 }
395
396
397
398 /************************** FUNZIONE PER CONTROLLARE
      LA SIMMETRIA ******************/
399
400 /************* Definizione: In matematica, una
      relazione binaria R in un insieme X   **/
401 /************* simmetrica se e solo se,presi due
      elementi qualsiasi a e b, vale che  **/
402 /************* se a   in relazione con b allora anche
      b   in relazione con a. *********/
403
404 int check_simmetria(struct relBin verifica){
405
406     int i,
407       j,
```

```
408       riscontro,
409       simmetria;
410
411     simmetria = 1;
412
413
414     i = 0;
415     j = 0;
416     riscontro = 0;
417
418   /*Check della simmetria per numeri*/
419
420     if(verifica.controllo == 1){
421
422       while( i < verifica.dimensione){
423
424         j = 0;
425         while( j < verifica.dimensione){
426
427           if(verifica.primo_termine[i] == verifica.
                secondo_termine[j])
428             if(verifica.primo_termine[j] == verifica.
                  secondo_termine[i])
429               riscontro++;
430
431           j++;
432       }
433
434         if(riscontro == 0){
435           j = verifica.dimensione;
436           i = verifica.dimensione;
437           simmetria = 0;
438         }
439         riscontro = 0;
440         i++;
441       }
442
443     }
444
445   /*Check della simmetria per stringhe*/
446
447     if(verifica.controllo == 2){
448
449       while( i < verifica.dimensione){
```

```
450
451         j = 0;
452         while( j < verifica.dimensione){
453
454           if(strcmp(verifica.prima_stringa[i],verifica.
                seconda_stringa[j]) == 0 )
455             if(strcmp(verifica.prima_stringa[j],verifica
                  .seconda_stringa[i]) == 0 )
456               riscontro++;
457
458           j++;
459         }
460
461         if(riscontro == 0){
462           j = verifica.dimensione;
463           i = verifica.dimensione;
464           simmetria = 0;
465         }
466         riscontro = 0;
467         i++;
468       }
469
470     }
471
472 /***** Controllo se la simmetria  stata verificata
       *********/
473
474     if(simmetria == 1)
475       printf("    e' simmetrica\n");
476     else
477       printf("    e' asimmetrica\n");
478
479 /****** Fine controllo simmetria ******/
480
481     return(simmetria);
482 }
483
484
485
486 /* FUNZIONE PER CONTROLLARE LA TRANSITIVIT */
487
488 /****** Definizione: In matematica, una relazione
       binaria R in un insieme X  transitiva se e solo se
```

```
489       per ogni a, b, c appartenenti ad X, se a   in
              relazione con b e b   in relazione con c,
              allora
490        a   in relazione con c.*******/
491
492
493   int check_transitivita(struct relBin verifica){
494
495     int i,
496       j,
497       k,
498       transitivita;
499
500   /*SETTO LA TRANSITIVITA INIZIALMENTE COME VERA E
        AZZERO I CONTATORI*/
501     transitivita = 1;
502     i = 0;
503     j = 0;
504     k = 0;
505
506   /*VERIFICA TRANSITIVIT PER NUMERI*/
507
508
509     if(verifica.controllo == 1){
510
511       while(i < verifica.dimensione){
512         j = 0;
513
514         while(j < verifica.dimensione){
515           k=0;
516
517           if(verifica.secondo_termine[i] == verifica.
                primo_termine[j]){
518             transitivita = 0;
519
520             while(k < verifica.dimensione){
521               if(verifica.primo_termine[i] == verifica.
                    primo_termine[k]){
522                 if(verifica.secondo_termine[k]==verifica
                      .secondo_termine[j]){
523                   transitivita = 1;
524                   j = verifica.dimensione;
525                   k = verifica.dimensione;
526                 }
```

```
527                 }
528
529                  k++;
530               }
531
532            }
533
534            j++;
535         }
536
537         i++;
538      }
539   }
540
541
542 /***************** VERIFICA TRANSITIVIT PER STRINGHE
      ***************/
543
544   if(verifica.controllo == 2){
545
546
547      while(i < verifica.dimensione){
548         j = 0;
549
550         while(j < verifica.dimensione){
551            k=0;
552
553            if(strcmp(verifica.seconda_stringa[i],verifica
                  .prima_stringa[j]) == 0){
554               transitivita = 0;
555
556               while(k < verifica.dimensione){
557                  if(strcmp(verifica.prima_stringa[i],
                        verifica.prima_stringa[k]) == 0){
558                     if(strcmp(verifica.seconda_stringa[k],
                           verifica.seconda_stringa[j]) == 0){
559                        transitivita = 1;
560                        j = verifica.dimensione;
561                        k = verifica.dimensione;
562                     }
563                  }
564
565                  k++;
566               }
```

```
567           }
568
569           j++;
570         }
571
572         i++;
573       }
574
575     }
576
577   /********** Controllo se la relazione    Transitiva
        *********/
578
579     if(transitivita == 1)
580       printf("   e' transitiva\n");
581
582     else
583       printf("   non e' transitiva\n");
584
585   /************ Fine controllo Transitivit ***********
        */
586
587     return(transitivita);
588
589   }
590
591   /********** Dicotomia ************/
592
593   int check_dicotomia(struct relBin verifica){
594
595     int a,b,c,d;
596     int numero_elementi;
597     int dicotomia = 0;
598     int dimensione;
599     int riscontro;
600     int secondo_riscontro;
601     a=0;
602     b=0;
603     c=0;
604     d=a-1;
605     dimensione = verifica.dimensione;
606
607   /********* Dicotomia per numeri *********/
608
```

```
609     if(verifica.controllo == 1){
610
611   /********** Conto il numero delle coppie esistenti (
          scarto le coppie uguali) *********/
612
613       while( a < verifica.dimensione){
614         d = a-1;
615         b = a+1;
616         secondo_riscontro = 0;
617
618       if(a>0){
619         while ( d >= 0 ){
620           if(verifica.primo_termine[a] == verifica.
                primo_termine[d]){
621             if(verifica.secondo_termine[a] == verifica.
                  secondo_termine[d])
622               secondo_riscontro = 1;
623           }
624           d--;
625         }
626       }
627
628       if(secondo_riscontro != 1){
629         while ( b < verifica.dimensione){
630           if(verifica.primo_termine[a] == verifica.
                primo_termine[b])
631             if(verifica.secondo_termine[a] == verifica.
                  secondo_termine[b]){
632               dimensione--;
633           }
634         b++;
635         }
636       }
637       a++;
638     }
639
640
641       a=0;
642       b=0;
643       c=0;
644       numero_elementi=0;
645       riscontro = 0;
646   /*************** Conto il numero degli elementi
          distinti esistenti ***************/
```

```
647
648       while(a<verifica.dimensione){
649         d=a-1;
650         secondo_riscontro = 0;
651
652         while(d >= 0){
653           if(verifica.primo_termine[a] == verifica.
                primo_termine[d])
654             secondo_riscontro = 1;
655             d--;
656         }
657         if(secondo_riscontro != 1){
658           if(verifica.primo_termine[a] == verifica.
                secondo_termine[a])
659             riscontro++;
660
661         }
662       a++;
663       }
664
665     numero_elementi = riscontro;
666     c = numero_elementi;
667
668   /************ Conto quanti dovrebbero essere gli
        elementi per avere la dicotomia ***********/
669
670     while(numero_elementi > 0){
671       numero_elementi--;
672       c = c + numero_elementi;
673       }
674     }
675
676   /******************** VERIFICA DICOTOMICA PER STRINGHE
        *****************/
677
678     if(verifica.controllo == 2){
679
680   /********** Conto il numero delle coppie esistenti (
        scarto le coppie uguali) *********/
681
682       while( a < verifica.dimensione){
683         d = a-1;
684         b = a+1;
685         secondo_riscontro = 0;
```

```
686       if(a>0){
687         while ( d >= 0 ){
688           if((strcmp(verifica.prima_stringa[a],verifica.
                prima_stringa[d])) == 0){
689             if((strcmp(verifica.seconda_stringa[a],
                  verifica.seconda_stringa[d])) == 0)
690               secondo_riscontro = 1;
691           }
692           d--;
693         }
694       }
695
696       if(secondo_riscontro != 1){
697         while ( b < verifica.dimensione){
698           if((strcmp(verifica.prima_stringa[a],verifica.
                prima_stringa[b])) == 0)
699             if((strcmp(verifica.seconda_stringa[a],
                  verifica.seconda_stringa[b])) == 0){
700               dimensione--;
701           }
702         b++;
703         }
704       }
705       a++;
706     }
707
708
709       a=0;
710       b=0;
711       c=0;
712       numero_elementi = 0;
713
714   /*************** Conto il numero degli elementi
        distinti esistenti ***************/
715
716       while(a<verifica.dimensione){
717         d=a-1;
718         secondo_riscontro = 0;
719
720         while(d >= 0){
721           if((strcmp(verifica.prima_stringa[a],verifica.
                prima_stringa[d])) == 0)
722             secondo_riscontro = 1;
723             d--;
```

```
724         }
725         if(secondo_riscontro != 1){
726           if((strcmp(verifica.prima_stringa[a],verifica.
                seconda_stringa[a])) == 0)
727             numero_elementi++;
728
729         }
730       a++;
731       }
732       c = numero_elementi;
733
734  /************ Conto quanti dovrebbero essere gli
       elementi per avere la dicotomia **********/
735
736       while(numero_elementi > 0){
737
738         numero_elementi--;
739         c = c + numero_elementi;
740
741       }
742
743     }
744
745  /************ Verifico se la dicotomia   verificata
       ***************/
746
747     if(dimensione == c)
748       dicotomia = 1;
749
750     if(dicotomia == 1 && (check_riflessivita(verifica)
         == 1))
751       printf("\t\t\te' dicotomica\n\n");
752
753     else
754       printf("\t\t\tnon e' dicotomica\n\n");
755
756  /***************** Fine verifica dicotomia
       *****************/
757
758     return(dicotomia);
759  }
760
761  /*Funzione di verifica dell'ordine totale*/
762
```

```
763
764 void ordine_totale (struct relBin verifica){
765
766   int parziale,
767     dicotomia;
768
769   parziale = ordine_parziale (verifica);
770   dicotomia = check_dicotomia (verifica);
771
772   if(parziale == 0)
773     printf("\n\n l'ordine non e' totale in quanto non e
          ' nemmeno parziale");
774
775   if(dicotomia == 0)
776     printf("\n\n l'ordine non e' totale in quanto non
          viene rispettata la propieta' di dicotomia");
777
778   if(dicotomia == 1 && parziale == 1)
779     printf("\n\nQuindi e' una relazione d'ordine totale
          ");
780
781   printf("\n\n\n\n....Controllo Ordine Totale Terminato
       ....\n\n\n\n");
782 }
783
784 /*Funzione che stabilisce se e' una relazione di
     equivalenza o meno*/
785
786 void relazione_equivalenza(struct relBin verifica){
787
788   int riflessivita;
789   int simmetria;
790   int transitivita;
791
792   riflessivita = check_riflessivita(verifica);
793   simmetria = check_simmetria(verifica);
794   transitivita = check_transitivita(verifica);
795
796   if(riflessivita == 1 && simmetria == 1 &&
        transitivita == 1)
797   printf("\n\nQuindi e' una relazione di equivalenza\n"
        );
798
799   if(riflessivita == 0)
```

```
800     printf("\n Quindi non e' una relazione di
            equivalenza perche' non riflessiva\n");
801
802     if(simmetria == 0)
803     printf("\n Quindi non e' una relazione di
            equivalenza perche' non simmetrica\n");
804
805     if(transitivita == 0)
806     printf("\n Quindi non e' una relazione di
            equivalenza perche' non transitiva\n");
807 }
808
809 /*Funzione che stabilisce se la relazione binaria
      acquisita e' una funzione matematica*/
810
811 void check_funzione(struct relBin verifica){
812
813   int i;
814   int k;
815   int termini_diversi;
816   int termini_uguali_prima;
817   int termini_uguali_dopo;
818   int errore;
819
820 if(verifica.controllo == 1){
821
822   i=0;
823   errore=0;
824   termini_diversi=0;
825   termini_uguali_dopo=0;
826   termini_uguali_prima=0;
827   while(i < verifica.dimensione){
828     k=verifica.dimensione-1;
829     termini_uguali_dopo=termini_uguali_prima;
830     while(k > i){
831       if(verifica.primo_termine[i] == verifica.
            primo_termine[k]){
832         if(verifica.secondo_termine[i] != verifica.
              secondo_termine[k]){
833           errore=1;
834           printf("\n Nel %d elemento c'e' un errore
                che impedisce alla realzione binaria\n",k
                +1);
835           printf("di essere una funzione\n");
```

```
836             k=i;
837             i=verifica.dimensione;
838           }
839           if(verifica.secondo_termine[i] == verifica.
                secondo_termine[k])
840             termini_uguali_dopo++;
841         }
842         k--;
843       }
844       if(errore == 0 && termini_uguali_dopo ==
             termini_uguali_prima)
845       termini_diversi++;
846
847       termini_uguali_prima = termini_uguali_dopo;
848       i++;
849     }
850     if(errore == 0 && (termini_diversi == (verifica.
          dimensione - termini_uguali_prima))){
851     printf("\n La relazione binaria e' una funzione\n");
852     check_biiettivita(verifica);
853   }
854     else
855     printf("\n La relazione binaria non e' una funzione\
          n");
856   }
857
858   /********** Controllo se c' una funzione per stringhe
        (le stringhe sono considerate come costanti di
        diverso valore) ***********/
859
860   if(verifica.controllo == 2){
861
862     i=0;
863     errore=0;
864     termini_diversi=0;
865     termini_uguali_dopo=0;
866     termini_uguali_prima=0;
867     while(i < verifica.dimensione){
868       k=verifica.dimensione-1;
869       termini_uguali_dopo=termini_uguali_prima;
870       while(k > i){
871         if((strcmp(verifica.prima_stringa[i],verifica.
              prima_stringa[k])) == 0){
```

```
872           if((strcmp(verifica.seconda_stringa[i],
                  verifica.seconda_stringa[k])) != 0){
873             errore=1;
874             printf("\n Nel %d elemento c'e' un errore
                    che impedisce alla realzione binaria\n",k
                    +1);
875             printf("di essere una funzione\n");
876             k=i;
877             i=verifica.dimensione;
878           }
879           else
880             termini_uguali_dopo++;
881         }
882         k--;
883       }
884       if(errore == 0 && termini_uguali_dopo ==
             termini_uguali_prima)
885       termini_diversi++;
886
887       termini_uguali_prima = termini_uguali_dopo;
888       i++;
889     }
890     if(errore == 0 && (termini_diversi == (verifica.
          dimensione - termini_uguali_prima))){
891     printf("\n La relazione binaria e' una funzione\n");
892     check_biiettivita(verifica);
893     }
894     else
895     printf("\n La relazione binaria non e' una funzione\
          n");
896 }
897
898 printf("\n\n .... Controllo Funzione Terminato ... \n\
       n\n\n");
899
900 }
901
902 /**********FUNZIONE PER IL CHECK DELL'INIETTIVITA
       '************/
903
904 int check_iniettivita(struct relBin verifica){
905
906   int i;
907   int k;
```

```
908     int termini_diversi;
909     int termini_uguali_prima;
910     int termini_uguali_dopo;
911     int errore;
912     int iniettivita;
913
914     iniettivita = 0;
915
916   if(verifica.controllo == 1){
917
918     i=0;
919     errore=0;
920     termini_diversi=0;
921     termini_uguali_dopo=0;
922     termini_uguali_prima=0;
923
924     while(i < verifica.dimensione){
925
926       k=verifica.dimensione-1;
927       termini_uguali_dopo=termini_uguali_prima;
928       while(k > i){
929
930         if(verifica.secondo_termine[i] == verifica.
              secondo_termine[k]){
931
932           if(verifica.primo_termine[i] != verifica.
                primo_termine[k]){
933
934             errore=1;
935             printf("\n Nel %d elemento c'e' un errore
                    che impedisce alla realzione binaria\n",k
                    +1);
936             printf("di essere una funzione\n");
937             k=i;
938             i=verifica.dimensione;
939           }
940           if(verifica.primo_termine[i] == verifica.
                primo_termine[k])
941             termini_uguali_dopo++;
942         }
943         k--;
944       }
945       if(errore == 0 && termini_uguali_dopo ==
            termini_uguali_prima)
```

```
946       termini_diversi++;
947
948       termini_uguali_prima = termini_uguali_dopo;
949       i++;
950     }
951     if(errore == 0 && (termini_diversi == (verifica.
            dimensione - termini_uguali_prima))){
952     printf("\n La relazione binaria e' iniettiva\n");
953     iniettivita = 1;
954   }
955     else
956     printf("\n La relazione binaria non e' iniettiva\n")
            ;
957
958
959   }
960
961   /********** Controllo iniettivita' per stringhe
          ***********/
962
963   if(verifica.controllo == 2){
964
965     i=0;
966     errore=0;
967     termini_diversi=0;
968     termini_uguali_dopo=0;
969     termini_uguali_prima=0;
970
971     while(i < verifica.dimensione){
972       k=verifica.dimensione-1;
973       termini_uguali_dopo=termini_uguali_prima;
974       while(k > i){
975         if((strcmp(verifica.seconda_stringa[i],verifica.
                seconda_stringa[k])) == 0){
976           if((strcmp(verifica.prima_stringa[i],verifica.
                  prima_stringa[k])) != 0){
977             errore=1;
978             printf("\n Nel %d elemento c'e' un errore
                    che impedisce alla realzione binaria\n",k
                    +1);
979             printf("di essere una funzione\n");
980             k=i;
981             i=verifica.dimensione;
982           }
```

```
983             if((strcmp(verifica.prima_stringa[i],verifica.
                    prima_stringa[k])) == 0)
984               termini_uguali_dopo++;
985           }
986
987           k--;
988         }
989         if(errore == 0 && termini_uguali_dopo ==
                termini_uguali_prima)
990         termini_diversi++;
991
992         termini_uguali_prima = termini_uguali_dopo;
993         i++;
994       }
995       if(errore == 0 && (termini_diversi == (verifica.
              dimensione - termini_uguali_prima))){
996       printf("\n La relazione binaria e' iniettiva");
997       iniettivita = 1;
998       }
999       else
1000      printf("\n La relazione binaria non e' iniettiva");
1001  }
1002
1003  return(iniettivita);
1004  }
1005
1006  /***************FUNZIONE PER IL CHECK DELLA
          SURIETTIVITA'************/
1007
1008  int check_suriettivita(struct relBin verifica){
1009
1010  /******* La suriettivit    sempre verificata in quanto
          il dominio e il codominio **********/
1011  /** sono entrambi i rispettivi x,y acquisiti, quindi
          non ho elementi y non associati a x **/
1012  int suriettivita;
1013
1014  suriettivita = 1;
1015  return(suriettivita);
1016  }
1017
1018  /***************FUNZIONE PER IL CHECK DELLA
          BIIETTIVITA'************/
1019
```

```
1020 void check_biiettivita(struct relBin verifica){
1021
1022   int    surriettivita,
1023       iniettivita;
1024
1025 surriettivita = check_suriettivita(verifica);
1026 iniettivita = check_iniettivita(verifica);
1027
1028
1029   if( surriettivita == 1 && iniettivita == 1)
1030     printf("\n la funzione e' biiettiva");
1031   else
1032     printf("\n la funzione non e' biiettiva");
1033 return;
1034 }
```

4.2 Test

```
1  #include<stdio.h>
2  #include"Progetto.h"
3
4  int main(void){
5    struct  relBin RelazioneBinaria;
6    int scelta;
7    int scan;
8
9    printf("\n Programma per effettuare i Test sulla
       libreria\n");
10
11
12   printf("\n\n Digita il numero corrispondente all'
       azione che si vuole svolgere\n");
13   printf("\n 1) Test Acquisizione\n 2) Esci\n");
14
15
16    while((scelta < 1) || (scelta > 2) || scan != 1){
17     printf("\n scelta: ");
18     fflush(stdin);
19     scan = scanf("%d",&scelta);
20   }
21   if(scelta == 1)
22   RelazioneBinaria = acquisizione(RelazioneBinaria);
23   if(scelta == 2){
24   printf("\n\n ..... Test terminati ......\n\n");
25   return(0);
26   }
27   scelta = -1;
28   while(scelta != 7){
29   printf("\n\n Digita il numero corrispondente all'
       azione che si vuole svolgere\n");
30   printf("\n 1) Test Acquisizione\n 2) Test Stampa\n
       3) Test verifica ordine parziale\n 4) Test
       verifica ordine totale");
31   printf("\n 5) Test verifica relazione d'equivalenza\
       n 6) Test funzione\n 7) Esci\n" );
32   scelta = -1;
33   while((scelta < 1) || (scelta > 7) || scan != 1){
34     printf("\n scelta: ");
35     fflush(stdin);
36     scan = scanf("%d",&scelta);
```

```
37    }
38
39
40    if(scelta == 1)
41      RelazioneBinaria = acquisizione(RelazioneBinaria);
42    if(scelta == 2)
43      stampa(RelazioneBinaria);
44    if(scelta == 3)
45      ordine_parziale(RelazioneBinaria);
46    if(scelta == 4)
47      ordine_totale(RelazioneBinaria);
48    if(scelta == 5)
49      relazione_equivalenza(RelazioneBinaria);
50    if(scelta == 6)
51      check_funzione(RelazioneBinaria);
52    if(scelta == 7){
53      printf("\n\n ..... Test terminati ..... \n\n");
54      return(0);
55      }
56    }
57  return(0);
58
59  }
```

4.3 Makefile

```
Test.exe: Test.c Makefile
gcc -ansi -Wall -O Test.c -o Test.exe
pulisci:
rm -f Test.o
pulisci_tutto:
rm -f Test.exe Test.o
```

5 Testing the program

Test 1:
Test di Relazione d′ ordine Totale.

Inputs: (a,a)(a,b)(b,b)

Outputs: checkriflessività : 1,checksimmetria : 0, checktransitività : 1 checkdicotomia : 1, la relazione è una relazione d′ ordine totale in quanto è rispetta anche la propietà di Dicotomia.

Test 2:
Test di Relazione d′ ordine Parziale.

Inputs:(a,a)(b,b)(a,b)(c,c)

Outputs:checkriflessività : 1,checksimmetria : 0, checktransitività : 1 la relazione è una relazione d′ ordine parziale in quanto rispetta le proprietà.

Test 3:
Test di Relazione d′ ordine non Parziale.

Inputs:(a,a)(b,b)(c,c)(d,d)(e,e)(a,b)(b,c)

Outputs:checkriflessività : 1,checksimmetria : 0, checktransitività : 0 la relazione non è una relazione d′ ordine parziale in quanto non rispetta le proprietà.

Test 4:
Test di Relazione d′ equivalenza.

Inputs:(a,a)(a,b)(b,a)

Outputs:checkriflessività : 1,checksimmetria : 1, checktransitività : 1 checkdicotomia : 0, la relazione è una relazione d′ equivalenza in quanto rispetta le proprietà.

Test 5:
Test di Relazione non d′ equivalenza.

Inputs:(a,a)(a,b)(b,c)

Outputs:checkriflessività : 0,checksimmetria : 0, checktransitività : 0 la relazione non è una relazione d′ ordine d′ equivalenza in quanto non rispetta le proprietà.

Test 6:
Test di Funzione.

Inputs:(a,a) Outputs:La relazione binaria e′ una funzione.
La relazione binaria e′ iniettiva.
La relazione binaria e′ biiettiva.

Test 7:
Test per verificare il controllo degli inputs.

Inputs:(casa rossa,casa blu)(casa blu,casa blu)(casa rossa,casa rossa)

Outputs:check_riflessività : 1,check_simmetria : 1, check_transitività : 1 dicotomia :1 la relazione è una relazione d′ ordine totale in quanto rispetta le proprietà.
le funzioni funzionano anche con input contenti degli spazi.

Test 8:
Test per inserire stringhe in una relazione numerica.

Inputs:(1,a)

Outputs: c′ è un′ errore reinserisci il valore.

stampa errore in quanto si era selezionato di voler immettere un input di tipo numerico.

Test 9:
Inputs:(1,2)(1,1)

Outputs: La relazione binaria non è una funzione

Test 10:
Inputs:(1,1)(2,1)

Outputs:La relazione binaria e$'$ una funzione
La funzione binaria non e$'$ iniettiva
La funzione binaria non e$'$ biiettiva

6 Verifying the program