Università di Urbino

Informatica Applicata

Programmazione Procedurale e Logica

Relazione

Progetto per la sessione invernale 2014/2015

Studente:
Marco Tamagno
matricola no: 261985

Studente:
Francesco Belacca
matricola no: 260492

Professore:
Marco Bernardo

May 29, 2015

Contents

1 Specifica del Problema

Scrivere un programma ANSI C che acquisisce da tastiera un insieme, una relazione binaria su quell'insieme ed un'operazione binaria su quell'insieme e poi verifica se l'insieme è chiuso rispetto all'operazione e se la relazione è una congruenza rispetto all'operazione.

2 Analisi del Problema

2.1 Input

Il problema prende in pasto come input un insieme, una relazione binaria su quell'insieme e un'operazione binaria su quell'insieme.

2.2 Output

Il problema ha come output il risultato della verifica della chiusura dell'insieme rispetto all'operazione e il risultato della verifica della congruenza della relazione rispetto all'operazione;

2.3 Relazioni tra input ed output

1)Chiusura:
Se due elementi qualsiasi, appartenenti all'insieme preso in considerazione vengono utilizzati come operandi per l'operazione immessa, si dice che l'operazione è chiusa rispetto all'insieme se e solo se anche il risultato dell'operazione appartiene all'insieme.

2)Congruenza:
Una relazione d'equivalenza su un insieme chiuso rispetto ad un'operazione è detta essere una congruenza rispetto a quell'operazione sse, ogni volta che si sostituisce un operando con un altro operando equivalente al primo, si ottiene un risultato equivalente a quello originario.

3 Progettazione dell'algoritmo

3.1 Scelte di progetto

La principale scelta di progetto è quella di restringere l'insieme degli input ai soli numeri.

3.2 Strutture utilizzate

I singoli elementi dell'insieme – acquisibili solo in modo sequenziale – debbono essere salvati in una struttura dati che agevoli la verifica delle proprietà. A tale scopo, risulta particolarmente adeguata una struttura dati che contenga un array unidimensionale e un intero che definisca quanti elementi sono stati acquisiti in totale. Chiameremo questa struttura Insieme, dato che è proprio ciò che deve rappresentare.
Per la relazione binaria invece, risulta più adeguata una struttura dati che contenga due array unidimensionali(uno contenete tutti i primi termini e uno tutti i secondi) insieme ad un altro intero che denoti il numero totale di coppie binarie acquisite. Chiameremo questa struttura relBin.
Infine per l'operazione, non c'è bisogno di salvare gli operandi, sapendo che devono appartenere all'insieme acquisito, perciò abbiamo deciso di chiedere all'utente ogni risultato delle operazioni possibili all'interno dell'insieme acquisito, in un semplice array unidimensionale, dicendogli di inserire 999 nel caso il risultato sia impossibile o indeterminato.

3.3 Passi del programma

-Acquisire e comunicare un insieme.
-Acquisire e comunicare una relazione binaria su quell'insieme.
-Acquisire e comunicare un operazione binaria su quell'insieme.
-Verificare e comunicare la chiusura dell'insieme rispetto all'operazione.
-Verificare e comunicare se la congruenza della relazione rispetto all'operazione.

4 Implementazione dell'algoritmo

Questa è la traduzione dei passi in C:

```
/************************************************/
/* Progetto per la sessione estiva del 2014/2015 */
/************************************************/

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

#include<stdio.h>
#include<stdlib.h>
#include<string.h>

/*********************************/
/* dichiarazione delle strutture */
/*********************************/
typedef struct Operazione
{
   double   *operando_a;
  double   *operando_b;
  double   *risultati;

} operazione_t;

typedef struct RelBin
{
   /* coppia numerica */

   double  *primo_termine;
   double *secondo_termine;

   /* variabile per sapere il numero delle coppie */

   int dimensione;
} rel_bin;

typedef struct Insieme
{
   double* elementi_insieme;
   int numero_elementi;
} insieme_t;

/*********************************/
/* dichiarazione delle funzioni */
/*********************************/

int controllo_simmetria (rel_bin);
```

```
47 int controllo_riflessivita (rel_bin);
48 int controllo_transitivita (rel_bin);
49 int relazione_equivalenza (rel_bin);
50 insieme_t acquisisci_insieme(void);
51 rel_bin acquisisci_rel_bin(insieme_t);
52 insieme_t crea_insieme_vuoto(void);
53 int acquisisci_elemento(insieme_t);
54 void stampa(rel_bin);
55 operazione_t acquisisci_operazione(insieme_t);
56 int controllo_chiusura(insieme_t,operazione_t);
57 void controllo_congruenza(rel_bin,insieme_t,operazione_t,int);
58
59 /*****************/
60 /* funzione main */
61 /*****************/
62
63 int main()
64 {
65   operazione_t operazione;
66   char carattere_non_letto;
67   int scelta;
68   int lettura_effettuata;
69   int ripeti;
70   int chiusura;
71
72     /* variabili per insieme e relazione */
73
74     insieme_t insieme;
75     rel_bin relazione;
76
77     /*inizializzo le variabili*/
78     ripeti = 0;
79     scelta = 0;
80     lettura_effettuata = 0;
81     chiusura = 1;
82
83
84     printf(" Questo programma acquisisce nel seguente ordine:\n");
85   printf("\n 1) Un insieme;\n 2) Una relazione binaria su");
86     printf(" quell'insieme;\n 3) Un'operazione binaria su quell");
87     printf("'insieme.\n\n Poi verifica se l'insieme e' chiuso ");
88     printf("rispetto all'operazione e se la relazione\n e' una");
89     printf(" congruenza rispetto all'operazione.\n");
90     printf("\n Digitare:\n 1 - se si vuole iniziare con");
91     printf(" l'acquisizione dell'insieme,\n 2 - se si vuole ");
92     printf("inserire l'insieme vuoto,");
93     printf("\n 3 - terminare il programma:");
94
```

```
95      while((scelta != 1 && scelta != 2 && scelta != 3) ||
            lettura_effettuata != 1)
96      {
97          lettura_effettuata = scanf("%d",&scelta);
98          if(lettura_effettuata != 1)
99          {
100             do
101                 carattere_non_letto = getchar();
102             while (carattere_non_letto != '\n');
103             scelta=4;
104         }
105         while(ripeti == 0 && lettura_effettuata == 1){
106           if(scelta==1)
107           {
108               insieme = acquisisci_insieme();
109               relazione = acquisisci_rel_bin(insieme);
110               stampa(relazione);
111         operazione = acquisisci_operazione(insieme);
112               chiusura = controllo_chiusura(insieme, operazione);
113               controllo_congruenza(relazione, insieme, operazione,
114                                    chiusura);
115           }
116           if(scelta==2){
117             insieme = crea_insieme_vuoto();
118         printf("\n L'insieme che si e' scelto e' vuoto, quindi ");
119         printf("qualsiasi \n sia la relazione, simmetria,");
120         printf(" riflessivita' e transitivita'");
121         printf("\n sono sempre verificate.");
122               printf("\n Per convenzione diciamo anche che qualsiasi
                      ");
123               printf(" sia l'operazione e' chiusa rispetto all'
                      insieme");
124       }
125           printf("\n Premere 0 per acquisire un altro insieme.\n ");
126           lettura_effettuata = scanf("%d",&ripeti);
127           if(lettura_effettuata != 1)
128           {
129               do
130                   carattere_non_letto = getchar();
131               while (carattere_non_letto != '\n');
132               ripeti = 1;
133           }
134         }
135
136
137     }
138     return 0;
139 }
140
```

```
141
142 /*****************************/
143 /* acquisizione dell'insieme */
144 /*****************************/
145
146 insieme_t acquisisci_insieme()
147 {
148    /*dichiaro la struttura insieme*/
149
150    insieme_t insieme;
151
152    int  i;     /*variabile contatore */
153    int  j;     /*variabile contatore*/
154    int  finisci_di_acquisire;/*variabile per terminare l'
           acquisizione*/
155    int  zeri;
156    int  elemento_acquisito; /*variabile per verificare che la
           acquisizione vada a buon fine*/
157
158    char carattere_non_letto; /*variabile necessaria allo
           svuotamento del buffer*/
159
160    double temporaneo;   /*variabile per acquisire ogni elemento
           temporaneamente*/
161
162    /*inizializzo le variabili*/
163
164    elemento_acquisito = 0;
165    j = 0;
166    i = 0;
167    zeri = 0;
168    temporaneo = 1;
169    insieme.numero_elementi = 50;
170    finisci_di_acquisire = 0;
171
172    /*alloco memoria*/
173    insieme.elementi_insieme = (double *) malloc (insieme.
           numero_elementi);
174
175    /*inizio la vera e propria acquisizione*/
176
177    printf("\n\n Si e' scelto di acquisire un'insieme\n");
178
179    /*chiedo se l'utente vuole inserire lo 0*/
180
181    printf("\n E' presente lo zero nell'insieme che si vuole
           inserire?");
182  printf("\n Inserire 1 per si altro per no:\n ");
183  do{
```

```
184     elemento_acquisito = scanf("%d",&zeri);
185         if(elemento_acquisito != 1)
186         {
187             do
188                 carattere_non_letto = getchar();
189             while (carattere_non_letto != '\n');
190         }
191   }while(elemento_acquisito != 1);
192     if (zeri == 1)
193     {
194         insieme.elementi_insieme = (double *) realloc (insieme.
              elementi_insieme, (i+1) * sizeof (double));
195         insieme.elementi_insieme[i]=0;
196         i=1;
197     }
198
199     /*faccio partire i da temporaneo*/
200
201     if(zeri != 1)
202     i=0;
203
204     printf("\n Per terminare l'acquisizione digitare 0\n");
205
206     while(finisci_di_acquisire != 1)
207     {
208
209         insieme.elementi_insieme = (double *) realloc (insieme.
              elementi_insieme, (i+1) * sizeof (double));
210         printf("\n Digitare ora il %d elemento: ",i+1);
211         elemento_acquisito = scanf("%lf",&temporaneo);
212
213     if(temporaneo == 0)
214         {
215             finisci_di_acquisire = 1;
216             insieme.numero_elementi = i;
217         }
218
219     if(i >= 0)
220             insieme.elementi_insieme[i] = temporaneo;
221
222         for(j = i - 1; j >= 0; j--){
223           if(elemento_acquisito != 1 || temporaneo == insieme.
                elementi_insieme[j])
224           {
225               do
226                   carattere_non_letto = getchar();
227               while (carattere_non_letto != '\n');
228         i--;
229         j = 0;
```

```
230         }
231
232       }
233       i++;
234    }
235
236
237    /**********************/
238    /* stampa dell'insieme */
239    /**********************/
240
241    printf("\n L'insieme acquisito e':");
242    printf("\n\n { ");
243    i=0;
244    while(i < insieme.numero_elementi)
245    {
246       printf("%.2lf",insieme.elementi_insieme[i]);
247       if(i+1 < insieme.numero_elementi)
248          printf(" ; ");
249       i++;
250    }
251    printf(" }");
252
253
254
255    return insieme;
256 }
257
258 insieme_t crea_insieme_vuoto()
259 {
260    insieme_t insieme;
261    insieme.elementi_insieme = (double *) malloc (1);
262    insieme.numero_elementi = 0;
263    return insieme;
264 }
265
266 rel_bin acquisisci_rel_bin(insieme_t insieme)
267 {
268    rel_bin relazione;
269
270    int acquisizione_finita,
271       risultato_lettura,
272       i,
273       primo_termine_acquisito;
274
275    char carattere_non_letto;
276
277    acquisizione_finita = 0;
278    primo_termine_acquisito = 0;
```

```
279
280     relazione.dimensione = 0;
281     relazione.primo_termine = (double *) malloc (2);
282     relazione.secondo_termine = (double *) malloc (2);
283
284     while (acquisizione_finita == 0)
285     {
286         primo_termine_acquisito = 0;
287         relazione.dimensione++;
288         acquisizione_finita = 2;
289
290         /*Acquisisco i termini della coppia*/
291
292         printf ("\n Inserisci i termini della coppia \n ");
293         relazione.primo_termine = (double *) realloc (relazione.
                primo_termine,
294                                 (relazione.dimensione+1) * sizeof (
                                    double));
295         relazione.secondo_termine = (double *) realloc (relazione.
                secondo_termine,
296                                   (relazione.dimensione+1) * sizeof (
                                      double));
297         risultato_lettura = 0;
298
299
300         /*Acquisisco il primo termine*/
301         if (primo_termine_acquisito == 0)
302         {
303             printf (" Primo Termine: ");
304             relazione.primo_termine[relazione.dimensione - 1] =
                    acquisisci_elemento(insieme);
305         }
306         primo_termine_acquisito = 1;
307
308         /*Acquisisco il secondo termine*/
309         if (primo_termine_acquisito == 1)
310         {
311             printf (" Secondo Termine: ");
312             relazione.secondo_termine[relazione.dimensione - 1] =
                    acquisisci_elemento(insieme);
313           for(i=relazione.dimensione-2;i>=0;i--)
314         if(relazione.primo_termine[relazione.dimensione - 1] ==
                relazione.primo_termine[i])
315           if(relazione.secondo_termine[relazione.dimensione -1] ==
                  relazione.secondo_termine[i]){
316             relazione.dimensione--;
317             i=0;
318           }
319     }
```

```
320
321         /*Chiedo all'utente se ci sono altre coppie*/
322
323         do
324         {
325             printf ("\n Vuoi acquisire un'altra coppia? immetti 1 per
                    uscire, 0 per continuare\n ");
326             printf ("\n scelta: ");
327             risultato_lettura = scanf ("%d",
328                                        &acquisizione_finita);
329             if (acquisizione_finita < 0 || acquisizione_finita > 1 ||
                    risultato_lettura != 1)
330                 do
331                     carattere_non_letto = getchar();
332                 while (carattere_non_letto != '\n');
333         }
334         while (acquisizione_finita < 0 || acquisizione_finita > 1 );
335     }
336     return relazione;
337 }
338
339 /*****************************FUNZIONE DI STAMPA
        *********************************/
340
341 void stampa (rel_bin stampa)
342 {
343
344     int i = 0;
345
346     printf ("\n La relazione binaria e':");
347     printf ("\n\n {");
348
349     /******Stampa per coppie numeriche *****/
350
351     while (i < stampa.dimensione)
352     {
353         printf ("(%.2lf,%.2lf)",stampa.primo_termine[i],stampa.
                secondo_termine[i]);
354         if (i+1 != stampa.dimensione)
355             printf (" ; ");
356         i++;
357     }
358     printf("}\n");
359     return ;
360 }
361
362 int acquisisci_elemento(insieme_t insieme)
363 {
364     /* dichiaro le variabili */
```

```
365     char carattere_non_letto;
366
367     int lettura_corretta,
368         i,
369         elemento_trovato;
370
371     double elemento;
372     /* inizializzo le variabili */
373     elemento=0;
374     lettura_corretta=1;
375     do
376     {
377         /* controllo che i valori siano stati letti correttamente */
378         /* e nel caso non sia cosi svuoto il buffer */
379         if(lettura_corretta != 1)
380         {
381             do
382                 carattere_non_letto = getchar();
383             while (carattere_non_letto != '\n');
384             printf ("\n C'e'un errore, reinserire il termine e
                    verificare\n");
385             printf(" che appartenga all'insieme precedentemente
                    inserito: \n ");
386         }
387         lettura_corretta = scanf("%lf",&elemento);
388         /* verifico se l'elemento che si vuole utilizzare nella
                relazione */
389         /* e' presente nell'insieme inserito */
390         elemento_trovato = 0;
391         for(i=0; i < insieme.numero_elementi; i++)
392             if(elemento == insieme.elementi_insieme[i])
393                 elemento_trovato = 1;
394
395         if(elemento_trovato == 0)
396             lettura_corretta = 0;
397     }
398     while(lettura_corretta == 0);
399
400     return elemento;
401 }
402
403
404 /* Acquisisco l'operazione*/
405
406 operazione_t acquisisci_operazione(insieme_t insieme){
407   operazione_t operazione;
408   int i,
409     j,
410     dimensione;
```

```
411   double *risultati;
412   i=0;
413   j=0;
414   dimensione=0;
415   operazione.risultati = (double *) malloc (2);
416   operazione.operando_a = (double *) malloc (2);
417   operazione.operando_b = (double *) malloc (2);
418   printf(" \n\n Inserire ora i risultati dell'operazioni: \n");
419   printf(" \n Digitare 999 per risultati impossibili o indeterminati
          . \n");
420   for(i = 0; i < insieme.numero_elementi; i++){
421         for(j = 0; j < insieme.numero_elementi; j++){
422           operazione.risultati = (double *) realloc (operazione.
                risultati, (dimensione+1) * sizeof (double));
423       operazione.operando_a = (double *) realloc (operazione.
              operando_a, (dimensione+1) * sizeof (double));
424       operazione.operando_b = (double *) realloc (operazione.
              operando_b, (dimensione+1) * sizeof (double));
425       operazione.operando_a[dimensione] = insieme.elementi_insieme[i
              ];
426       operazione.operando_b[dimensione] = insieme.elementi_insieme[j
              ];
427       printf("\n %f * %f = ",insieme.elementi_insieme[i],insieme.
              elementi_insieme[j]);
428       scanf("%lf",&operazione.risultati[dimensione]);
429       dimensione++;
430     }
431   }
432   return operazione;
433 }
434
435 int controllo_chiusura(insieme_t insieme,operazione_t operazione){
436 int i,
437   j,
438   chiusura;
439 i=0;
440 j=0;
441 chiusura=0;
442 for(i=0;i<(insieme.numero_elementi*insieme.numero_elementi);i++){
443     chiusura = 0;
444   if(operazione.risultati[i] != 999)
445   for(j=0;j<insieme.numero_elementi;j++)
446     if(operazione.risultati[i] == insieme.elementi_insieme[j]){
447     chiusura = 1;
448     j = insieme.numero_elementi+1;
449     }
450   if(chiusura == 0)
451   i=(insieme.numero_elementi*insieme.numero_elementi);
452   }
```

```
453
454   if(chiusura == 0)
455   printf("\n La chiusura non e' verificata\n");
456   if(chiusura == 1)
457   printf("\n La chiusura e' verificata\n");
458
459   return chiusura;
460 }
461
462 int controllo_riflessivita (rel_bin verifica)
463 {
464
465     int i,
466         j,
467         k,
468         riscontro,
469         secondo_riscontro,
470         riflessivita;
471
472     riflessivita = 1;
473     i = 0;
474     j = 0;
475     k = 0;
476     riscontro = 0;
477     secondo_riscontro = 0;
478
479     /*Verifica riflessivit*/
480
481     /*Definizione: una relazione per la quale esiste almeno un
            elemento che non e'in relazione con s stesso non soddisfa la
            definizione di riflessivit*/
482
483     while ( (i < verifica.dimensione) && (k < verifica.dimensione))
484     {
485
486         /*Verifica riflessivit per numeri*/
487
488             riscontro = 0;
489             secondo_riscontro = 0;
490             if (verifica.primo_termine[i] == verifica.secondo_termine
                    [i])
491                 riscontro++; /****Controllo se c' stato un riscontro a
                        ,a****/
492             secondo_riscontro++;
493             if (riscontro != 0)
494             {
495                 i++;
496                 k++;
497             }
```

```
498             /**/
499             else
500             {
501                 j=0;
502                 riscontro = 0;
503                 secondo_riscontro = 0;
504
505                 /***************** Controllo la riflessivit per gli
                        elementi del primo insieme
                        ******************************/
506
507                 while (j < verifica.dimensione)
508                 {
509                     if (j == i)
510                         j++;
511                     else
512                     {
513                         if (verifica.primo_termine[i] == verifica.
                                primo_termine[j])
514                             if (verifica.primo_termine[j] == verifica.
                                    secondo_termine[j])
515                                 riscontro++;
516
517                         j++;
518                     }
519                 }
520
521                 j = 0;
522
523                 /***************** Controllo la riflessivit per gli
                        elementi del secondo insieme
                        ******************************/
524
525                 while (j < verifica.dimensione)
526                 {
527                     if (j == k)
528                         j++;
529                     else
530                     {
531                         if (verifica.secondo_termine[k] == verifica.
                                secondo_termine[j])
532                             if (verifica.primo_termine[j] == verifica.
                                    secondo_termine[j])
533                                 secondo_riscontro++;
534
535                         j++;
536                     }
537                 }
538                 if (riscontro != 0)
```

```
539                 i++;
540
541             /**** Se non c' stato un riscontro di riflessivit esco
                     e imposto la riflessivit a 0 *****/
542
543             else
544             {
545                 i=verifica.dimensione;
546                 riflessivita = 0;
547             }
548
549             if (secondo_riscontro != 0)
550                 k++;
551
552             else
553             {
554                 k=verifica.dimensione;
555                 riflessivita = 0;
556             }
557           }
558
559  }
560
561  /********* Controllo se  riflessiva ******************/
562
563    if (riflessivita == 1)
564       printf (" e'riflessiva\n");
565    else
566       printf (" non e'riflessiva\n");
567
568    /*********** Fine riflessivita **********************/
569  return (riflessivita);
570 }
571
572 int controllo_transitivita (rel_bin verifica)
573 {
574
575    int i,
576       j,
577       k,
578       transitivita;
579
580    /*IMPOSTO LA TRANSITIVITA INIZIALMENTE COME VERA E AZZERO I
       CONTATORI*/
581    transitivita = 1;
582    i = 0;
583    j = 0;
584    k = 0;
585
```

```
586     /*VERIFICA TRANSITIVIT PER NUMERI*/
587
588
589         while (i < verifica.dimensione)
590         {
591             j = 0;
592
593             while (j < verifica.dimensione)
594             {
595                 k=0;
596
597                 if (verifica.secondo_termine[i] == verifica.
                        primo_termine[j])
598                 {
599                     transitivita = 0;
600
601                     while (k < verifica.dimensione)
602                     {
603                         if (verifica.primo_termine[i] == verifica.
                                primo_termine[k])
604                         {
605                             if (verifica.secondo_termine[k]==verifica.
                                    secondo_termine[j])
606                             {
607                                 transitivita = 1;
608                                 k = verifica.dimensione;
609                             }
610                         }
611
612                         k++;
613                     }
614
615                     if (transitivita==0)
616                     {
617                         j=verifica.dimensione;
618                         i=verifica.dimensione;
619                     }
620                 }
621
622                 j++;
623             }
624
625             i++;
626         }
627
628
629
630     /********** Controllo se la relazione  Transitiva *********/
631
```

```
632     if (transitivita == 1)
633         printf (" e'transitiva\n");
634 
635     else
636         printf (" non e'transitiva\n");
637 
638     /************ Fine controllo Transitivit ************/
639 
640     return (transitivita);
641 
642 }
643 
644 
645 int relazione_equivalenza (rel_bin verifica)
646 {
647 
648     int riflessivita,
649       simmetria,
650       transitivita,
651       equivalenza;
652 
653   equivalenza=0;
654     riflessivita = controllo_riflessivita(verifica);
655     simmetria = controllo_simmetria(verifica);
656     transitivita = controllo_transitivita(verifica);
657 
658     if (riflessivita == 1 && simmetria == 1 && transitivita == 1){
659         printf ("\n Quindi e'una relazione di equivalenza\n");
660     equivalenza=1;
661   }
662 
663     if (riflessivita == 0)
664         printf ("\n Quindi non e'una relazione di equivalenza perche'
                non riflessiva\n");
665 
666     if (simmetria == 0)
667         printf ("\n Quindi non e'una relazione di equivalenza perche'
                non simmetrica\n");
668 
669     if (transitivita == 0)
670         printf ("\n Quindi non e'una relazione di equivalenza perche'
                non transitiva\n");
671     return equivalenza;
672 }
673 
674 int controllo_simmetria (rel_bin verifica)
675 {
676 
677     int i,
```

```
678         j,
679         riscontro,
680         simmetria;
681
682     simmetria = 1;
683
684
685     i = 0;
686     j = 0;
687     riscontro = 0;
688
689     /*controllo della simmetria per numeri*/
690
691         while ( i < verifica.dimensione)
692         {
693
694             j = 0;
695             while ( j < verifica.dimensione)
696             {
697
698                 if (verifica.primo_termine[i] == verifica.
                        secondo_termine[j])
699                     if (verifica.primo_termine[j] == verifica.
                            secondo_termine[i])
700                         riscontro++;
701                 j++;
702             }
703
704             if (riscontro == 0)
705             {
706                 j = verifica.dimensione;
707                 i = verifica.dimensione;
708                 simmetria = 0;
709             }
710             riscontro = 0;
711             i++;
712         }
713     /**** Controllo se la simmetria  stata verificata ****/
714         if (simmetria == 1)
715             printf (" e'simmetrica\n");
716         else
717             printf (" e'asimmetrica\n");
718
719
720     return (simmetria);
721 }
722
723
```

```
724 void controllo_congruenza(rel_bin relazione, insieme_t insieme,
        operazione_t operazione,int chiusura)
725 {
726 int equivalenza,
727   controllo,
728   i,
729   j,
730   k;
731
732 equivalenza = relazione_equivalenza(relazione);
733
734 i = 0;
735 j = 0;
736 k = 0;
737 controllo=1;
738
739 for(i=0;i<relazione.dimensione;i++)
740 {
741   for(j=0;j<(insieme.numero_elementi*insieme.numero_elementi);j++)
742   {
743     if(relazione.primo_termine[i] == operazione.operando_a[j])
744       for(k=0;k<(insieme.numero_elementi*insieme.numero_elementi);k
              ++)
745         if(relazione.secondo_termine[i] == operazione.operando_a[k]
              && operazione.operando_b[j] == operazione.operando_b[k])
746           if(operazione.risultati[j] != operazione.risultati[k])
747           {
748             controllo = 0;
749             k=(insieme.numero_elementi*insieme.numero_elementi);
750             j=(insieme.numero_elementi*insieme.numero_elementi);
751             i=relazione.dimensione;
752           }
753     if(relazione.primo_termine[i] == operazione.operando_b[j])
754       for(k=0;k<(insieme.numero_elementi*insieme.numero_elementi);k
              ++)
755         if(relazione.secondo_termine[i] == operazione.operando_b[k]
              && operazione.operando_a[j] == operazione.operando_a[k])
756           if(operazione.risultati[j] != operazione.risultati[k])
757           {
758             controllo = 0;
759             k=(insieme.numero_elementi*insieme.numero_elementi);
760             j=(insieme.numero_elementi*insieme.numero_elementi);
761             i=relazione.dimensione;
762           }
763   }
764 }
765
766
767   if(equivalenza == 0 || controllo == 0 || chiusura == 0)
```

```
768   printf("\n La cogruenza non e' verificata\n");
769   else
770   printf("\n La congruenza e' verificata\n");
771
772   return;
773 }
```