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 17, 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:

```
1 /***********************************************/
2 /* Progetto per la sessione estiva del 2014/2015 */
3 /***********************************************/
4
5 /*****************************/
6 /* inclusione delle librerie */
```

```
7  /*****************************/
8
9  #include<stdio.h>
10 #include<stdlib.h>
11 #include<string.h>
12
13 /*******************************/
14 /* dichiarazione delle strutture */
15 /*******************************/
16
17 typedef struct relBin
18 {
19     /* coppia numerica */
20
21     double  *primo_termine,
22             *secondo_termine;
23
24     /* variabile per sapere il numero delle coppie */
25
26     int dimensione;
27 } rel_bin;
28
29 typedef struct Insieme
30 {
31     double* elementi_insieme;
32     int numero_elementi;
33 } insieme_t;
34
35 /*******************************/
36 /* dichiarazione delle funzioni */
37 /*******************************/
38
39 int controllo_simmetria (rel_bin);
40 int controllo_riflessivita (rel_bin);
41 int controllo_transitivita (rel_bin);
42 int relazione_equivalenza (rel_bin);
43 insieme_t acquisisci_insieme(void);
44 rel_bin acquisisci_rel_bin(insieme_t);
45 insieme_t crea_insieme_vuoto(void);
46 int acquisisci_elemento(insieme_t);
47 void stampa(rel_bin);
48 double* acquisisci_operazione(insieme_t);
49 void controllo_chiusura(insieme_t,double *);
50 int controllo(insieme_t, double);
```

```
51 void controllo_congruenza(rel_bin,insieme_t,double *);
52
53 /*****************/
54 /* funzione main */
55 /*****************/
56
57 int main()
58 {
59     char carattere_non_letto;
60     double *risultati;
61   int scelta,
62         lettura_effettuata,
63         ripeti;
64
65     /* variabili per insieme e relazione */
66
67     insieme_t insieme;
68     rel_bin relazione;
69
70     /*inizializzo le variabili*/
71     ripeti = 0;
72     scelta=0;
73     lettura_effettuata=0;
74
75
76     printf("␣Questo␣programma␣acquisisce␣nel␣seguente␣
          ordine:␣\n\n␣1)␣Un␣insieme;␣");
77     printf("␣\n␣2)␣Una␣relazione␣binaria␣su␣quell'
          insieme;␣\n␣3)␣Un'operazione␣binaria");
78     printf("␣su␣quell'insieme.\n\n␣Poi␣verifica␣se␣l'
          insieme␣e'␣chiuso␣rispetto␣all'operazione");
79     printf("␣e␣se␣la␣relazione\n␣e'␣una␣congruenza␣
          rispetto␣all'operazione.\n");
80     printf("\n␣Digitare:\n␣1␣-␣se␣si␣vuole␣iniziare␣
          con␣l'acquisizione␣dell'insieme,\n␣2␣-␣se␣si␣
          vuole");
81     printf("␣inserire␣l'insieme␣vuoto,\n␣3␣-␣terminare
          ␣il␣programma:␣");
82     while((scelta != 1 && scelta != 2 && scelta != 3)
          || lettura_effettuata != 1)
83     {
84         lettura_effettuata = scanf("%d",&scelta);
85         if(lettura_effettuata != 1)
86         {
```

```
87              do
88                  carattere_non_letto = getchar();
89              while (carattere_non_letto != '\n');
90              scelta=4;
91          }
92          while(ripeti == 0){
93            if(scelta==1)
94            {
95                insieme = acquisisci_insieme();
96                relazione = acquisisci_rel_bin(insieme);
97                stampa(relazione);
98          risultati = acquisisci_operazione(insieme);
99                controllo_chiusura(insieme, risultati);
100               controllo_congruenza(relazione, insieme,
                      risultati);
101           }
102           if(scelta==2){
103             insieme = crea_insieme_vuoto();
104         printf("\n L'insieme che si e' scelto e' vuoto
              , quindi non ci possono essere");
105         printf("\n relazioni.\n");
106         risultati = acquisisci_operazione(insieme);
107               controllo_chiusura(insieme, risultati);
108               controllo_congruenza(relazione, insieme,
                      risultati);
109       }
110           printf("\n Premere 0 per acquisire un altro
                 insieme.\n ");
111           lettura_effettuata = scanf("%d",&ripeti);
112           if(lettura_effettuata != 1)
113           {
114               do
115                   carattere_non_letto = getchar();
116               while (carattere_non_letto != '\n');
117               ripeti = 1;
118           }
119         }
120
121
122     }
123     return 0;
124 }
125
126
```

```
127 /****************************/
128 /* acquisizione dell'insieme */
129 /****************************/
130
131 insieme_t acquisisci_insieme()
132 {
133     /*dichiaro la struttura insieme*/
134     insieme_t insieme;
135
136     int    i,              /*variabile contatore*/
137            finisci_di_acquisire,       /*variabile
                per terminare l'acquisizione*/
138            zeri,
139            elemento_acquisito;     /*variabile per
                verificare che la acquisizione vada a
                buon fine*/
140
141     char  carattere_non_letto;    /*variabile
            necessaria allo svuotamento del buffer*/
142
143     double temporaneo;          /*variabile per
            acquisire ogni elemento temporaneamente*/
144     /*inizializzo le variabili*/
145
146     elemento_acquisito = 0;
147     i = 0;
148     zeri = 0;
149     temporaneo = 1;
150     insieme.numero_elementi = 50;
151     finisci_di_acquisire = 0;
152
153     /*alloco memoria*/
154     insieme.elementi_insieme = (double *) malloc (
            insieme.numero_elementi);
155
156     /*inizio la vera e propria acquisizione*/
157
158     printf("\n\n Si e' scelto di acquisire un'insieme"
            );
159
160     /*chiedo se l'utente vuole inserire lo 0*/
161
162     printf("\n Premere il numero di zeri presenti nell
            'insieme: ");
```

```
163   do{
164     elemento_acquisito = scanf("%d",&zeri);
165         if(elemento_acquisito != 1)
166         {
167             do
168                 carattere_non_letto = getchar();
169             while (carattere_non_letto != '\n');
170         }
171   }while(elemento_acquisito != 1);
172     while (i < zeri && zeri > 0)
173     {
174         insieme.elementi_insieme = (double *) realloc
              (insieme.elementi_insieme, (i+1) * sizeof (
              double));
175         insieme.elementi_insieme[i]=0;
176         i++;
177     }
178
179     /*faccio partire i da temporaneo*/
180
181     i=zeri;
182
183     printf("\n Per terminare l'acquisizione digitare
          0\n");
184     while(finisci_di_acquisire != 1)
185     {
186         insieme.elementi_insieme = (double *) realloc
              (insieme.elementi_insieme, (i+1) * sizeof (
              double));
187         printf("\n Digitare ora il %d elemento: ",i+1)
              ;
188         elemento_acquisito = scanf("%lf",&temporaneo);
189
190         if(elemento_acquisito != 1)
191         {
192             do
193                 carattere_non_letto = getchar();
194             while (carattere_non_letto != '\n');
195       i--;
196         }
197         if(temporaneo == 0)
198         {
199             finisci_di_acquisire = 1;
200             insieme.numero_elementi = i;
```

```
201             }
202             else
203         if(i >= 0)
204                 insieme.elementi_insieme[i] = temporaneo;
205             i++;
206         }
207
208
209         /**********************/
210         /* stampa dell'insieme */
211         /**********************/
212
213         printf("\n L'insieme acquisito e':");
214         printf("\n\n {");
215         i=0;
216         while(i < insieme.numero_elementi)
217         {
218             printf("%.2lf",insieme.elementi_insieme[i]);
219             if(i+1 < insieme.numero_elementi)
220                 printf(" ; ");
221             i++;
222         }
223         printf(" }");
224
225
226
227         return insieme;
228     }
229
230     insieme_t crea_insieme_vuoto()
231     {
232         insieme_t insieme;
233         insieme.elementi_insieme = (double *) malloc (1);
234         insieme.numero_elementi = 0;
235         return insieme;
236     }
237
238     rel_bin acquisisci_rel_bin(insieme_t insieme)
239     {
240         rel_bin relazione;
241
242         int acquisizione_finita,
243             risultato_lettura,
244             primo_termine_acquisito;
```

```
245
246     char carattere_non_letto;
247
248     acquisizione_finita = 0;
249     primo_termine_acquisito = 0;
250
251     relazione.dimensione = 0;
252     relazione.primo_termine = (double *) malloc (2);
253     relazione.secondo_termine = (double *) malloc (2);
254     while (acquisizione_finita == 0)
255     {
256         primo_termine_acquisito = 0;
257         relazione.dimensione++;
258         acquisizione_finita = 2;
259
260         /*Acquisisco i termini della coppia*/
261
262         printf ("\n Inserisci i termini della coppia \
              n ");
263         relazione.primo_termine = (double *) realloc (
              relazione.primo_termine,
264                                     (relazione.
                                        dimensione+1) *
                                        sizeof (double));
265         relazione.secondo_termine = (double *) realloc
              (relazione.secondo_termine,
266                                       (relazione.
                                          dimensione+1) *
                                           sizeof (double
                                          ));
267         risultato_lettura = 0;
268
269
270         /*Acquisisco il primo termine*/
271         if (primo_termine_acquisito == 0)
272         {
273             printf ("  Primo Termine: ");
274             relazione.primo_termine[relazione.
                  dimensione - 1] = acquisisci_elemento(
                  insieme);
275         }
276         primo_termine_acquisito = 1;
277
278         /*Acquisisco il secondo termine*/
```

```
279             if (primo_termine_acquisito == 1)
280             {
281                 printf ("   Secondo Termine: ");
282                 relazione.secondo_termine[relazione.
                        dimensione - 1] = acquisisci_elemento(
                        insieme);
283             }
284
285             /*Chiedo all'utente se ci sono altre coppie*/
286
287             do
288             {
289                 printf ("\n Vuoi acquisire un'altra coppia
                        ? immetti 1 per uscire , 0 per
                        continuare\n ");
290                 printf ("\n scelta: ");
291                 risultato_lettura = scanf ("%d",
292                                            &
                                                acquisizione_finita
                                                );
293                 if (acquisizione_finita < 0 ||
                        acquisizione_finita > 1 ||
                        risultato_lettura != 1)
294                     do
295                         carattere_non_letto = getchar();
296                     while (carattere_non_letto != '\n');
297             }
298             while (acquisizione_finita < 0 ||
                    acquisizione_finita > 1 );
299         }
300         return relazione;
301 }
302
303 /*****************************FUNZIONE DI STAMPA
        ******************************/
304
305 void stampa (rel_bin stampa)
306 {
307
308     int i = 0;
309
310     printf ("\n La relazione binaria e ':");
311     printf ("\n\n   {");
312
```

```
313     /******Stampa per coppie numeriche *****/
314
315     while (i < stampa.dimensione)
316     {
317         printf ("(%.2lf,%.2lf)",stampa.primo_termine[i
                ],stampa.secondo_termine[i]);
318         if (i+1 != stampa.dimensione)
319             printf (" ; ");
320         i++;
321     }
322     printf("}\n");
323     return ;
324 }
325
326 int acquisisci_elemento(insieme_t insieme)
327 {
328     /* dichiaro le variabili */
329     char carattere_non_letto;
330
331     int lettura_corretta,
332         i,
333         elemento_trovato;
334
335     double elemento;
336     /* inizializzo le variabili */
337     elemento=0;
338     lettura_corretta=1;
339     do
340     {
341         /* controllo che i valori siano stati letti
                correttamente */
342         /* e nel caso non sia cosi svuoto il buffer */
343         if(lettura_corretta != 1)
344         {
345             do
346                 carattere_non_letto = getchar();
347             while (carattere_non_letto != '\n');
348             printf ("\n C'e'un errore, reinserire il
                termine e verificare\n");
349             printf(" che appartenga all'insieme
                precedentemente inserito: \n ");
350         }
351         lettura_corretta = scanf("%lf",&elemento);
```

```
352         /* verifico se l'elemento che si vuole
                utilizzare nella relazione */
353         /* e' presente nell'insieme inserito */
354         elemento_trovato = 0;
355         for(i=0; i < insieme.numero_elementi; i++)
356             if(elemento == insieme.elementi_insieme[i
                    ])
357                 elemento_trovato = 1;
358
359         if(elemento_trovato == 0)
360             lettura_corretta = 0;
361     }
362     while(lettura_corretta == 0);
363
364     return elemento;
365 }
366
367
368 /* Acquisisco l'operazione*/
369
370 double* acquisisci_operazione(insieme_t insieme){
371   int i,
372     j,
373     dimensione;
374   double *risultati;
375   i=0;
376   j=0;
377   dimensione=0;
378   risultati = (double *) malloc (2);
379   printf("\n\nInserire ora i risultati dell'
        operazioni:\n");
380   printf("\nDigitare 999 per risultati impossibili o
        indeterminati.\n");
381   for(i = 0; i < insieme.numero_elementi; i++){
382         for(j = 0; j < insieme.numero_elementi; j++){
383           risultati = (double *) realloc (risultati, (
                dimensione+1) * sizeof (double));
384       printf("\n%f * %f =",insieme.elementi_insieme[
            i],insieme.elementi_insieme[j]);
385       scanf("%lf",&risultati[dimensione]);
386       dimensione++;
387     }
388   }
389   return risultati;
```

```
390 }
391
392 void controllo_chiusura(insieme_t insieme,double *
      risultati){
393 int i,
394   j,
395   chiusura;
396 i=0;
397 j=0;
398 chiusura=0;
399 for(i=0;i<(insieme.numero_elementi*insieme.
      numero_elementi);i++){
400     chiusura = 0;
401   if(risultati[i] != 999)
402   for(j=0;j<insieme.numero_elementi;j++)
403     if(risultati[i] == insieme.elementi_insieme[j]){
404     chiusura = 1;
405     j = insieme.numero_elementi+1;
406     }
407   if(chiusura == 0)
408   i=(insieme.numero_elementi*insieme.numero_elementi);
409   }
410
411   if(chiusura == 0)
412   printf("\n La chiusura non e' verificata\n");
413   if(chiusura == 1)
414   printf("\n La chiusura e' verificata\n");
415
416   return;
417 }
418
419 int controllo_riflessivita (rel_bin verifica)
420 {
421
422     int i,
423         j,
424         k,
425         riscontro,
426         secondo_riscontro,
427         riflessivita;
428
429     riflessivita = 1;
430     i = 0;
431     j = 0;
```

```
432     k = 0;
433     riscontro = 0;
434     secondo_riscontro = 0;
435
436     /*Verifica àriflessivit*/
437
438     /*Definizione: una relazione per la quale esiste
            almeno un elemento che non e'in relazione con és
             stesso non soddisfa la definizione di
            àriflessivit*/
439
440     while ( (i < verifica.dimensione) && (k < verifica
            .dimensione))
441     {
442
443         /*Verifica àriflessivit per numeri*/
444
445             riscontro = 0;
446             secondo_riscontro = 0;
447             if (verifica.primo_termine[i] == verifica.
                    secondo_termine[i])
448                 riscontro++; /****Controllo se cè'
                        stato un riscontro a,a****/
449             secondo_riscontro++;
450             if (riscontro != 0)
451             {
452                 i++;
453                 k++;
454             }
455             /**/
456             else
457             {
458                 j=0;
459                 riscontro = 0;
460                 secondo_riscontro = 0;
461
462                 /***************** Controllo la
                        àriflessivit per gli elementi del
                        primo insieme
                        ****************************/
463
464                 while (j < verifica.dimensione)
465                 {
466                     if (j == i)
```

```
467                     j++;
468                 else
469                 {
470                     if (verifica.primo_termine[i]
                        == verifica.primo_termine[j
                        ])
471                         if (verifica.primo_termine
                            [j] == verifica.
                            secondo_termine[j])
472                             riscontro++;
473
474                     j++;
475                 }
476             }
477
478             j = 0;
479
480             /**************** Controllo la
                àriflessivit per gli elementi del
                secondo insieme
                ****************************/
481
482             while (j < verifica.dimensione)
483             {
484                 if (j == k)
485                     j++;
486                 else
487                 {
488                     if (verifica.secondo_termine[k
                        ] == verifica.
                        secondo_termine[j])
489                         if (verifica.primo_termine
                            [j] == verifica.
                            secondo_termine[j])
490                             secondo_riscontro++;
491
492                     j++;
493                 }
494             }
495             if (riscontro != 0)
496                 i++;
497
498             /**** Se non cè' stato un riscontro di
                àriflessivit esco e imposto la
```

```
                    à riflessivit a 0 *****/
499
500                 else
501                 {
502                     i=verifica.dimensione;
503                     riflessivita = 0;
504                 }
505
506                 if (secondo_riscontro != 0)
507                     k++;
508
509                 else
510                 {
511                     k=verifica.dimensione;
512                     riflessivita = 0;
513                 }
514             }
515
516   }
517
518   /********* Controllo se è riflessiva
          ******************/
519
520     if (riflessivita == 1)
521         printf ("___e'riflessiva\n");
522     else
523         printf ("___non_e'riflessiva\n");
524
525     /********** Fine riflessivita
          **********************/
526   return (riflessivita);
527 }
528
529 int controllo_transitivita (rel_bin verifica)
530 {
531
532     int i,
533         j,
534         k,
535         transitivita;
536
537     /*IMPOSTO LA TRANSITIVITA INIZIALMENTE COME VERA E
          AZZERO I CONTATORI*/
538     transitivita = 1;
```

```
539         i = 0;
540         j = 0;
541         k = 0;
542
543         /*VERIFICA àTRANSITIVIT PER NUMERI*/
544
545
546             while (i < verifica.dimensione)
547             {
548                 j = 0;
549
550                 while (j < verifica.dimensione)
551                 {
552                     k=0;
553
554                     if (verifica.secondo_termine[i] ==
                        verifica.primo_termine[j])
555                     {
556                         transitivita = 0;
557
558                         while (k < verifica.dimensione)
559                         {
560                             if (verifica.primo_termine[i]
                                == verifica.primo_termine[k
                                ])
561                             {
562                                 if (verifica.
                                    secondo_termine[k]==
                                    verifica.
                                    secondo_termine[j])
563                                 {
564                                     transitivita = 1;
565                                     k = verifica.
                                        dimensione;
566                                 }
567                             }
568
569                             k++;
570                         }
571
572                         if (transitivita==0)
573                         {
574                             j=verifica.dimensione;
575                             i=verifica.dimensione;
```

```
576                         }
577                     }
578
579                     j++;
580                 }
581
582                 i++;
583             }
584
585
586
587         /********** Controllo se la relazione è Transitiva
                *********/
588
589         if (transitivita == 1)
590             printf ("___e'transitiva\n");
591
592         else
593             printf ("___non_e'transitiva\n");
594
595         /************ Fine controllo àTransitivit
                ************/
596
597         return (transitivita);
598
599 }
600
601
602 int relazione_equivalenza (rel_bin verifica)
603 {
604
605         int riflessivita,
606           simmetria,
607           transitivita,
608           equivalenza;
609
610       equivalenza=0;
611         riflessivita = controllo_riflessivita(verifica);
612         simmetria = controllo_simmetria(verifica);
613         transitivita = controllo_transitivita(verifica);
614
615         if (riflessivita == 1 && simmetria == 1 &&
              transitivita == 1){
```

```
616             printf ("\n Quindi e' una relazione di
                    equivalenza\n");
617         equivalenza=1;
618       }
619
620         if (riflessivita == 0)
621             printf ("\n Quindi non e' una relazione di
                    equivalenza perche' non riflessiva\n");
622
623         if (simmetria == 0)
624             printf ("\n Quindi non e' una relazione di
                    equivalenza perche' non simmetrica\n");
625
626         if (transitivita == 0)
627             printf ("\n Quindi non e' una relazione di
                    equivalenza perche' non transitiva\n");
628         return equivalenza;
629     }
630
631     int controllo_simmetria (rel_bin verifica)
632     {
633
634         int i,
635             j,
636             riscontro,
637             simmetria;
638
639         simmetria = 1;
640
641
642         i = 0;
643         j = 0;
644         riscontro = 0;
645
646         /*controllo della simmetria per numeri*/
647
648             while ( i < verifica.dimensione)
649             {
650
651                 j = 0;
652                 while ( j < verifica.dimensione)
653                 {
654
```

```
655                     if (verifica.primo_termine[i] ==
                            verifica.secondo_termine[j])
656                         if (verifica.primo_termine[j] ==
                                verifica.secondo_termine[i])
657                             riscontro++;
658                     j++;
659                 }
660
661                 if (riscontro == 0)
662                 {
663                     j = verifica.dimensione;
664                     i = verifica.dimensione;
665                     simmetria = 0;
666                 }
667                 riscontro = 0;
668                 i++;
669             }
670         /**** Controllo se la simmetria è stata verificata
                ****/
671             if (simmetria == 1)
672                 printf ("    e'simmetrica\n");
673             else
674                 printf ("    e'asimmetrica\n");
675
676
677         return (simmetria);
678     }
679
680
681     void controllo_congruenza(rel_bin relazione, insieme_t
            insieme, double * risultati)
682     {
683     int equivalenza,
684       i,
685       j,
686       continua;
687
688     equivalenza = relazione_equivalenza(relazione);
689     i = 0;
690     j = 0;
691     continua=0;
692     for(i=0;i<insieme.numero_elementi;i++){
693         for(j=0; j<(insieme.numero_elementi*insieme.
                numero_elementi); j++)
```

```
694         if(insieme.elementi_insieme[j] == risultati[i])
695       continua = 1;
696       j = (insieme.numero_elementi*insieme.
              numero_elementi)+1;
697     if(continua == 0)
698     i=insieme.numero_elementi +1;
699     }
700
701     if(continua == 0 || equivalenza == 0)
702     printf("\n La cogruenza non e' verificata\n");
703     else
704     printf("\n La congruenza e' verificata\n");
705
706     return;
707
708 }
```