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 31, 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

4.1 Programma

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 typedef struct Operazione
17 {
18   double   *operando_a;
19   double   *operando_b;
20   double   *risultati;
21
22 } operazione_t;
23
24 typedef struct RelBin
25 {
26   /* coppia numerica */
27
28   double  *primo_termine;
29   double *secondo_termine;
30
31   /* variabile per sapere il numero delle coppie */
32
33   int dimensione;
34 } rel_bin;
35
36 typedef struct Insieme
37 {
38   double* elementi_insieme;
39   int numero_elementi;
40 } insieme_t;
41
42 /*********************************/
43 /* dichiarazione delle funzioni */
44 /*********************************/
```

```
45
46  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 = 1;
79    scelta = 0;
80    lettura_effettuata = 0;
81    chiusura = 1;
82
83    while(ripeti == 1)
84    {
85      printf("\n ************************************");
86      printf("************************\n");
87      printf("\n Questo programma acquisisce nel seguente");
88      printf(" ordine:\n");
89      printf("\n 1) Un insieme;\n 2) Una relazione binaria su ");
90      printf("quell'insieme;\n 3) Un'operazione binaria su quell");
91      printf("'insieme.\n\n Poi verifica se l'insieme e' chiuso ");
92      printf("rispetto all'operazione \n ");
93      printf(" e se la relazione e' una");
```

```
94    printf(" congruenza rispetto all'operazione.\n");
95    printf("\n ********************************");
96    printf("*****************************\n");
97    printf("\n\n Digitare:\n 1 - se si vuole iniziare con");
98    printf(" l'acquisizione dell'insieme,\n 2 - se si vuole ");
99    printf("inserire l'insieme vuoto,");
100   printf("\n 3 - terminare il programma: ");
101
102   do
103   {
104     lettura_effettuata = scanf("%d",&scelta);
105     if(lettura_effettuata != 1)
106     {
107       do
108         carattere_non_letto = getchar();
109       while (carattere_non_letto != '\n');
110       scelta=4;
111     }
112   }
113   while((scelta != 1 && scelta != 2
114         && scelta != 3) || lettura_effettuata != 1);
115
116   if(scelta == 1)
117   {
118     insieme = acquisisci_insieme();
119     relazione = acquisisci_rel_bin(insieme);
120     stampa(relazione);
121     operazione = acquisisci_operazione(insieme);
122     chiusura = controllo_chiusura(insieme, operazione);
123     controllo_congruenza(relazione, insieme, operazione,
124                          chiusura);
125   }
126   if(scelta == 2)
127   {
128     printf("\n\n ************** INSIEME");
129     printf("VUOTO **********************\n");
130     insieme = crea_insieme_vuoto();
131     printf("\n L'insieme che si e' scelto e' vuoto,");
132     printf(" quindi qualsiasi \n sia la relazione");
133     printf(", simmetria, riflessivita' e transitivita'\n");
134     printf(" sono sempre verificate.\n Per convenzione ");
135     printf("diciamo anche che qualsiasi sia\n l'operazione");
136     printf(" e' chiusa rispetto all'insieme");
137   }
138
139   printf("\n\n Digitare:\n 1 - se si vuole acquisire");
140   printf(" un altro insieme,\n 2 - se si vuole uscire: ");
141
142   do
```

```
143     {
144       lettura_effettuata = scanf("%d",&ripeti);
145       if(lettura_effettuata != 1)
146       {
147         do
148           carattere_non_letto = getchar();
149         while (carattere_non_letto != '\n');
150         ripeti = 1;
151       }
152     }
153     while(lettura_effettuata != 1 || (ripeti != 1 && ripeti != 2));
154   }
155 
156   return 0;
157 }
158 
159 
160 /*****************************/
161 /* acquisizione dell'insieme */
162 /*****************************/
163 
164 insieme_t acquisisci_insieme()
165 {
166   /*dichiaro la struttura insieme*/
167 
168   insieme_t insieme;
169 
170   /*variabile contatore */
171   int  i;
172   /*variabile contatore*/
173   int  j;
174   /*variabile per terminare l'acquisizione*/
175   int  finisci_di_acquisire;
176   /*variabile per l'acquisizione dell'elemento 0*/
177   int  zeri;
178   /*variabile per verificare che la
179   acquisizione vada a buon fine*/
180   int  elemento_acquisito;
181   /*variabile necessaria allo
182   svuotamento del buffer*/
183   char carattere_non_letto;
184   /*variabile per acquisire ogni
185   elemento temporaneamente*/
186   double temporaneo;
187 
188   /*inizializzo le variabili*/
189 
190   elemento_acquisito = 0;
191   j = 0;
```

```
192   i = 0;
193   zeri = 0;
194   temporaneo = 1;
195   insieme.numero_elementi = 50;
196   finisci_di_acquisire = 0;
197
198   /*alloco memoria*/
199   insieme.elementi_insieme = (double *)
200                           malloc (insieme.numero_elementi);
201
202   /*inizio la vera e propria acquisizione*/
203
204   printf("\n\n Si e' scelto di acquisire un'insieme\n");
205
206   /*chiedo se l'utente vuole inserire lo 0*/
207
208   printf("\n\n ************** ACQUISIZIONE DELL'");
209   printf("INSIEME **********************\n");
210   printf("\n\n Digitare:\n 1 - se l'elemento 0");
211   printf(" appartiene all'insieme");
212   printf("\n 2 - nel caso non gli appartiene: ");
213
214   do
215   {
216     elemento_acquisito = scanf("%d",&zeri);
217     if(elemento_acquisito != 1)
218     {
219       do
220         carattere_non_letto = getchar();
221       while (carattere_non_letto != '\n');
222     }
223   }
224   while(elemento_acquisito != 1 || (zeri != 1 && zeri != 2));
225
226   if (zeri == 1)
227   {
228     insieme.elementi_insieme = (double *)
229                             realloc (insieme.elementi_insieme,
230                                      (i+1) * sizeof (double));
231     insieme.elementi_insieme[i] = 0;
232     i = 1;
233   }
234
235   /*faccio partire i i+1 se c'e' lo zero*/
236
237   if(zeri == 2)
238     i = 0;
239
240   printf("\n\n Per terminare l'acquisizione digitare 0\n\n");
```

```
241
242   while(finisci_di_acquisire != 1)
243   {
244     insieme.elementi_insieme = (double *)
245                             realloc (insieme.elementi_insieme,
246                                     (i+1) * sizeof (double));
247     printf("\n Digitare ora il %d elemento: ",i+1);
248     elemento_acquisito = scanf("%lf",&temporaneo);
249
250     if(temporaneo == 0)
251     {
252       finisci_di_acquisire = 1;
253       insieme.numero_elementi = i;
254     }
255
256     if(i >= 0)
257       insieme.elementi_insieme[i] = temporaneo;
258
259     for(j = i - 1; j >= 0; j--)
260     {
261       if(elemento_acquisito != 1 ||
262          temporaneo == insieme.elementi_insieme[j])
263       {
264         do
265           carattere_non_letto = getchar();
266         while (carattere_non_letto != '\n');
267         i--;
268         j = 0;
269       }
270
271     }
272     i++;
273   }
274
275
276   /***********************/
277   /* stampa dell'insieme */
278   /***********************/
279   printf("\n\n ************** STAMPA DELL'");
280   printf("INSIEME **************************\n");
281   printf("\n\n L'insieme acquisito e':");
282   printf("\n\n { ");
283   i=0;
284
285   while(i < insieme.numero_elementi)
286   {
287     printf("%.2lf",insieme.elementi_insieme[i]);
288     if(i+1 < insieme.numero_elementi)
289       printf(" ; ");
```

```
290     i++;
291   }
292   printf(" }\n\n");
293
294
295
296   return insieme;
297 }
298
299 insieme_t crea_insieme_vuoto()
300 {
301   insieme_t insieme;
302   insieme.elementi_insieme = (double *) malloc (1);
303   insieme.numero_elementi = 0;
304   return insieme;
305 }
306
307 rel_bin acquisisci_rel_bin(insieme_t insieme)
308 {
309   printf("\n\n ********* ACQUISIZIONE DELLA");
310   printf("RELAZIONE BINARIA ***************\n");
311   rel_bin relazione;
312
313   int acquisizione_finita,
314       risultato_lettura,
315       i,
316       primo_termine_acquisito;
317
318   char carattere_non_letto;
319
320   acquisizione_finita = 1;
321   primo_termine_acquisito = 0;
322
323   relazione.dimensione = 0;
324   relazione.primo_termine = (double *) malloc (2);
325   relazione.secondo_termine = (double *) malloc (2);
326
327   while (acquisizione_finita == 1)
328   {
329     primo_termine_acquisito = 0;
330     relazione.dimensione++;
331     acquisizione_finita = 2;
332
333     /*Acquisisco i termini della coppia*/
334
335     printf ("\n\n Inserisci i termini della coppia \n ");
336
337     relazione.primo_termine = (double *)
338                               realloc (relazione.primo_termine,
```

```
                                    (relazione.dimensione+1)
                                    * sizeof (double));

  relazione.secondo_termine = (double *)
                            realloc (relazione.secondo_termine,
                                     (relazione.dimensione+1)
                                     * sizeof (double));
  risultato_lettura = 0;


  /*Acquisisco il primo termine*/
  if (primo_termine_acquisito == 0)
  {
    printf (" Primo Termine: ");
    relazione.primo_termine[relazione.dimensione - 1] =
      acquisisci_elemento(insieme);
  }
  primo_termine_acquisito = 1;

  /*Acquisisco il secondo termine*/
  if (primo_termine_acquisito == 1)
  {
    printf (" Secondo Termine: ");
    relazione.secondo_termine[relazione.dimensione - 1]
      = acquisisci_elemento(insieme);

    for(i=relazione.dimensione-2; i>=0; i--)
      if(relazione.primo_termine[relazione.dimensione - 1]
          == relazione.primo_termine[i])
        if(relazione.secondo_termine[relazione.dimensione -1]
            == relazione.secondo_termine[i])
        {
          relazione.dimensione--;
          i = 0;
        }
  }

  /*Chiedo all'utente se ci sono altre coppie*/

  do
  {
    printf("\n\n Digitare:\n 0 - per");
    printf("terminare l'acquisizione,");
    printf("\n 1 - se si vuole acquisire un altra coppia: ");
    risultato_lettura = scanf ("%d",
                               &acquisizione_finita);
    if (acquisizione_finita < 0 ||
        acquisizione_finita > 1 || risultato_lettura != 1)
      do
```

```
388           carattere_non_letto = getchar();
389         while (carattere_non_letto != '\n');
390     }
391     while (acquisizione_finita < 0 || acquisizione_finita > 1 );
392   }
393
394   return relazione;
395 }
396
397 /************FUNZIONE DI STAMPA****************/
398
399 void stampa (rel_bin stampa)
400 {
401
402   int i = 0;
403   printf("\n\n ********* STAMPA DELLA RELAZIONE BINARIA *****");
404   printf ("****************\n\n La relazione binaria e':");
405   printf ("\n\n {");
406
407   /******Stampa per coppie numeriche *****/
408
409   while (i < stampa.dimensione)
410   {
411     printf ("(%.2lf,%.2lf)",
412             stampa.primo_termine[i],
413             stampa.secondo_termine[i]);
414     if (i+1 != stampa.dimensione)
415       printf (" ; ");
416     i++;
417   }
418   printf("}\n");
419   return;
420 }
421
422 int acquisisci_elemento(insieme_t insieme)
423 {
424   /* dichiaro le variabili */
425   char carattere_non_letto;
426
427   int lettura_corretta,
428       i,
429       elemento_trovato;
430
431   double elemento;
432   /* inizializzo le variabili */
433   elemento = 0;
434   lettura_corretta = 1;
435
436   do
```

```
437   {
438     /* controllo che i valori siano
439         stati letti correttamente
440      e nel caso svuoto il buffer */
441
442     if(lettura_corretta != 1)
443     {
444       do
445         carattere_non_letto = getchar();
446       while (carattere_non_letto != '\n');
447       printf ("\n C'e'un errore, reinserire ");
448       printf ("il termine e verificare\n");
449       printf (" che appartenga all'insieme");
450       printf ("precedentemente inserito: \n ");
451     }
452     lettura_corretta = scanf("%lf",&elemento);
453
454     /* verifico se l'elemento che si
455        vuole utilizzare nella relazione
456      e' presente nell'insieme inserito */
457
458     elemento_trovato = 0;
459
460     for(i=0; i < insieme.numero_elementi; i++)
461       if(elemento == insieme.elementi_insieme[i])
462         elemento_trovato = 1;
463
464     if(elemento_trovato == 0)
465       lettura_corretta = 0;
466   }
467   while(lettura_corretta == 0);
468
469   return elemento;
470 }
471
472
473 /* Acquisisco l'operazione*/
474
475 operazione_t acquisisci_operazione(insieme_t insieme)
476 {
477   operazione_t operazione;
478   char carattere_non_letto;
479   int i,
480       j,
481       dimensione,
482       controllo;
483
484   i = 0;
485   j = 0;
```

```
486   dimensione = 0;
487
488   operazione.risultati = (double *) malloc (2);
489   operazione.operando_a = (double *) malloc (2);
490   operazione.operando_b = (double *) malloc (2);
491   printf("\n\n ********* ACQUISIZIONE ");
492   printf("DELL'OPERAZIONE **********************\n");
493   printf(" \n\n Inserire ora i risultati dell'operazioni: \n");
494   printf(" \n Digitare 999 per risultati ");
495   printf("impossibili o indeterminati. \n");
496
497   for(i = 0; i < insieme.numero_elementi; i++)
498   {
499     for(j = 0; j < insieme.numero_elementi; j++)
500     {
501       operazione.risultati = (double *)
502                           realloc (operazione.risultati,
503                                   (dimensione+1)
504                                   * sizeof (double));
505       operazione.operando_a = (double *)
506                            realloc (operazione.operando_a,
507                                    (dimensione+1)
508                                    * sizeof (double));
509       operazione.operando_b = (double *)
510                            realloc (operazione.operando_b,
511                                    (dimensione+1)
512                                    * sizeof (double));
513       operazione.operando_a[dimensione] = insieme.elementi_insieme[i
              ];
514       operazione.operando_b[dimensione] = insieme.elementi_insieme[j
              ];
515       printf("\n %f * %f = ",insieme.elementi_insieme[i],
516             insieme.elementi_insieme[j]);
517       do
518       {
519
520         controllo = scanf("%lf",
521                         &operazione.risultati[dimensione]);
522         if(controllo != 1)
523         {
524           do
525             carattere_non_letto = getchar();
526           while (carattere_non_letto != '\n');
527         }
528       }
529
530       while(controllo != 1);
531       dimensione++;
532     }
```

```
533   }
534   return operazione;
535 }
536
537 int controllo_chiusura(insieme_t insieme,operazione_t operazione)
538 {
539   int i,
540       j,
541       chiusura;
542
543   i = 0;
544   j = 0;
545   chiusura = 0;
546
547   for(i = 0;
548       i<(insieme.numero_elementi*insieme.numero_elementi);
549       i++)
550   {
551     chiusura = 0;
552     if(operazione.risultati[i] != 999)
553       for(j=0; j<insieme.numero_elementi; j++)
554         if(operazione.risultati[i] ==
555             insieme.elementi_insieme[j])
556         {
557           chiusura = 1;
558           j = insieme.numero_elementi+1;
559         }
560     if(chiusura == 0)
561       i = (insieme.numero_elementi*insieme.numero_elementi);
562   }
563   printf("\n\n ************ CHIUSURA **********************");
564   printf("******************\n");
565   if(chiusura == 0)
566     printf("\n\n La chiusura non e' verificata\n");
567   if(chiusura == 1)
568     printf("\n\n La chiusura e' verificata\n");
569
570   return chiusura;
571 }
572
573 int controllo_riflessivita (rel_bin verifica)
574 {
575
576   int i,
577       j,
578       k,
579       riscontro,
580       secondo_riscontro,
581       riflessivita;
```

```
582
583   riflessivita = 1;
584   i = 0;
585   j = 0;
586   k = 0;
587   riscontro = 0;
588   secondo_riscontro = 0;
589
590   /*Verifica riflessivita'*/
591
592   /*Definizione: una relazione per la quale
593    esiste almeno un elemento che non e'in relazione
594   con se' stesso non soddisfa la definizione di riflessivita'*/
595
596   while ( (i < verifica.dimensione) && (k < verifica.dimensione))
597   {
598
599     /*Verifica riflessivita' per numeri*/
600
601     riscontro = 0;
602     secondo_riscontro = 0;
603     if (verifica.primo_termine[i] == verifica.secondo_termine[i])
604       riscontro++;/*Controllo se c'e' stato un riscontro a,a*/
605     secondo_riscontro++;
606     if (riscontro != 0)
607     {
608       i++;
609       k++;
610     }
611     /**/
612     else
613     {
614       j = 0;
615       riscontro = 0;
616       secondo_riscontro = 0;
617
618       /* Controllo la riflessivita' per
619     gli elementi del primo insieme */
620
621       while (j < verifica.dimensione)
622       {
623         if (j == i)
624           j++;
625         else
626         {
627           if (verifica.primo_termine[i] ==
628               verifica.primo_termine[j])
629             if (verifica.primo_termine[j] ==
630                 verifica.secondo_termine[j])
```

```
631             riscontro++;
632
633           j++;
634         }
635       }
636
637       j = 0;
638
639       /*Controllo la riflessivita' per gli
640     elementi del secondo insieme*/
641
642       while (j < verifica.dimensione)
643       {
644         if (j == k)
645           j++;
646         else
647         {
648           if (verifica.secondo_termine[k] ==
649               verifica.secondo_termine[j])
650             if (verifica.primo_termine[j] ==
651                 verifica.secondo_termine[j])
652               secondo_riscontro++;
653
654           j++;
655         }
656       }
657       if (riscontro != 0)
658         i++;
659
660       /**** Se non c'e' stato un riscontro di riflessivita'
661        esco e imposto la riflessivita' a 0 *****/
662
663       else
664       {
665         i = verifica.dimensione;
666         riflessivita = 0;
667       }
668
669       if (secondo_riscontro != 0)
670         k++;
671
672       else
673       {
674         k = verifica.dimensione;
675         riflessivita = 0;
676       }
677     }
678
679   }
```

```
680
681
682   /*********** Fine riflessivita **********************/
683   return (riflessivita);
684 }
685
686 int controllo_transitivita (rel_bin verifica)
687 {
688
689   int i,
690       j,
691       k,
692       transitivita;
693
694   /*IMPOSTO LA TRANSITIVITA' INIZIALMENTE COME VERA
695    E AZZERO I CONTATORI*/
696
697   transitivita = 1;
698   i = 0;
699   j = 0;
700   k = 0;
701
702   /*VERIFICA TRANSITIVITa' PER NUMERI*/
703
704
705   while (i < verifica.dimensione)
706   {
707     j = 0;
708
709     while (j < verifica.dimensione)
710     {
711       k = 0;
712
713       if (verifica.secondo_termine[i] ==
714           verifica.primo_termine[j])
715       {
716         transitivita = 0;
717
718         while (k < verifica.dimensione)
719         {
720           if (verifica.primo_termine[i] ==
721               verifica.primo_termine[k])
722           {
723             if (verifica.secondo_termine[k] ==
724                 verifica.secondo_termine[j])
725             {
726               transitivita = 1;
727               k = verifica.dimensione;
728             }
```

```
729           }
730
731           k++;
732         }
733
734         if (transitivita==0)
735         {
736           j = verifica.dimensione;
737           i = verifica.dimensione;
738         }
739       }
740
741       j++;
742     }
743
744     i++;
745   }
746
747   /************ Fine controllo Transitivita' ************/
748
749   return (transitivita);
750
751 }
752
753
754 int relazione_equivalenza (rel_bin verifica)
755 {
756
757   int riflessivita,
758       simmetria,
759       transitivita,
760       equivalenza;
761
762   equivalenza = 0;
763   riflessivita = controllo_riflessivita(verifica);
764   simmetria = controllo_simmetria(verifica);
765   transitivita = controllo_transitivita(verifica);
766
767   if (riflessivita == 1 && simmetria == 1 && transitivita == 1)
768   {
769     printf ("\n e' una relazione di equivalenza\n");
770     equivalenza=1;
771   }
772
773   if (riflessivita == 0)
774   {
775     printf ("\n non e'una relazione di equivalenza ");
776     printf ("perche' non e' riflessiva\n");
777   }
```

```
778   if (simmetria == 0)
779   {
780     printf ("\n non e'una relazione di equivalenza ");
781     printf ("perche' non e' simmetrica\n");
782   }
783   if (transitivita == 0)
784   {
785     printf ("\n non e'una relazione di equivalenza ");
786     printf ("perche' non e' transitiva\n");
787   }
788   return equivalenza;
789 }
790
791 int controllo_simmetria (rel_bin verifica)
792 {
793
794   int i,
795       j,
796       riscontro,
797       simmetria;
798
799   simmetria = 1;
800
801
802   i = 0;
803   j = 0;
804   riscontro = 0;
805
806   /*controllo della simmetria per numeri*/
807
808   while ( i < verifica.dimensione)
809   {
810
811     j = 0;
812     while ( j < verifica.dimensione)
813     {
814
815       if (verifica.primo_termine[i] ==
816           verifica.secondo_termine[j])
817         if (verifica.primo_termine[j] ==
818             verifica.secondo_termine[i])
819           riscontro++;
820       j++;
821     }
822
823     if (riscontro == 0)
824     {
825       j = verifica.dimensione;
826       i = verifica.dimensione;
```

```
827       simmetria = 0;
828     }
829     riscontro = 0;
830     i++;
831   }
832
833   return (simmetria);
834 }
835
836
837 void controllo_congruenza(rel_bin relazione,
838                           insieme_t insieme,
839                           operazione_t operazione,
840                           int chiusura)
841 {
842   printf("\n\n ********* CONTROLLO LA CONGRUENZA");
843   printf(" ***************************\n");
844   int equivalenza,
845       controllo,
846       i,
847       j,
848       k;
849
850   equivalenza = relazione_equivalenza(relazione);
851
852   i = 0;
853   j = 0;
854   k = 0;
855   controllo=1;
856
857   for(i = 0; i<relazione.dimensione; i++)
858   {
859     for(j=0;
860         j<(insieme.numero_elementi*insieme.numero_elementi);
861         j++)
862     {
863       if(relazione.primo_termine[i] ==
864          operazione.operando_a[j])
865         for(k = 0;
866             k<(insieme.numero_elementi*insieme.numero_elementi);
867             k++)
868           if(relazione.secondo_termine[i] ==
869              operazione.operando_a[k] &&
870              operazione.operando_b[j] ==
871              operazione.operando_b[k])
872
873             if(operazione.risultati[j]
874                != operazione.risultati[k])
875             {
```

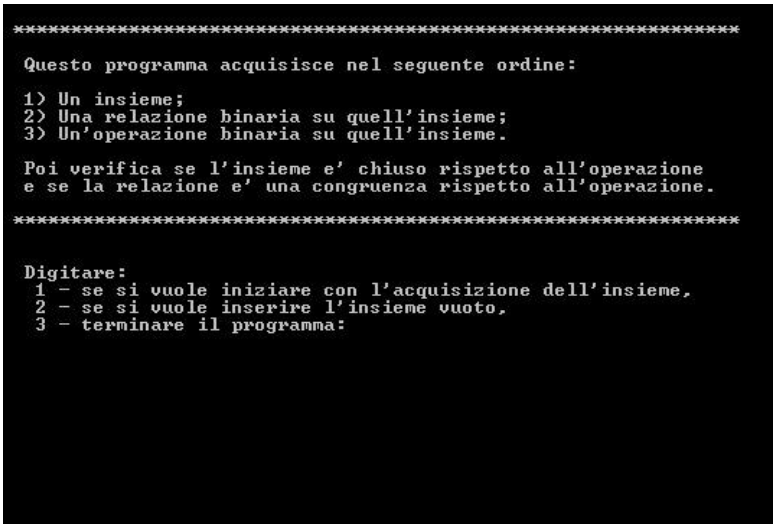
```
876             controllo = 0;
877             k = (insieme.numero_elementi*
878                  insieme.numero_elementi);
879
880             j = (insieme.numero_elementi*
881                  insieme.numero_elementi);
882
883             i = relazione.dimensione;
884           }
885       if(relazione.primo_termine[i] ==
886          operazione.operando_b[j])
887         for(k = 0;
888             k < insieme.numero_elementi*insieme.numero_elementi;
889             k++)
890           if(relazione.secondo_termine[i] ==
891              operazione.operando_b[k] &&
892              operazione.operando_a[j] ==
893              operazione.operando_a[k])
894
895             if(operazione.risultati[j] !=
896                operazione.risultati[k])
897             {
898               controllo = 0;
899               k = insieme.numero_elementi*
900                    insieme.numero_elementi;
901
902               j = insieme.numero_elementi*
903                    insieme.numero_elementi;
904
905               i = relazione.dimensione;
906             }
907     }
908   }
909
910
911   if(equivalenza == 0 || controllo == 0 || chiusura == 0)
912     printf("\n\n La cogruenza non e' verificata\n");
913   else
914     printf("\n\n La congruenza e' verificata\n");
915
916   return;
917 }
```

4.2 Makefile

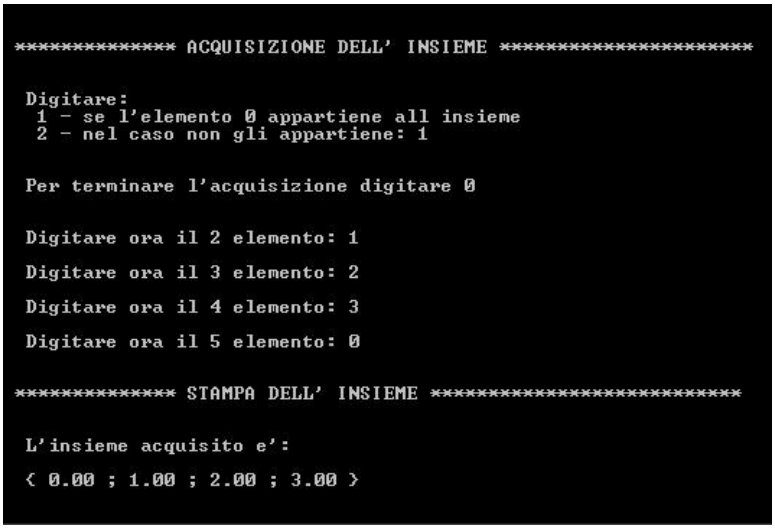
```
1 Progetto_sessione_estiva_PPL: Progetto_sessione_estiva_PPL.c
      Makefile
2   gcc -ansi -Wall -O Progetto_sessione_estiva_PPL.c -o
        Progetto_sessione_estiva_PPL
3 pulisci:
4   rm -f Progetto_sessione_estiva_PPL.o
5 pulisci_tutto:
6   rm -f Progetto_sessione_estiva_PPL Progetto_sessione_estiva_PPL.o
```

5 Testing del programma

Spiego all'utente cosa fa il programma..



Acquisisco l'insieme e lo stampo per farlo vedere all'utente..



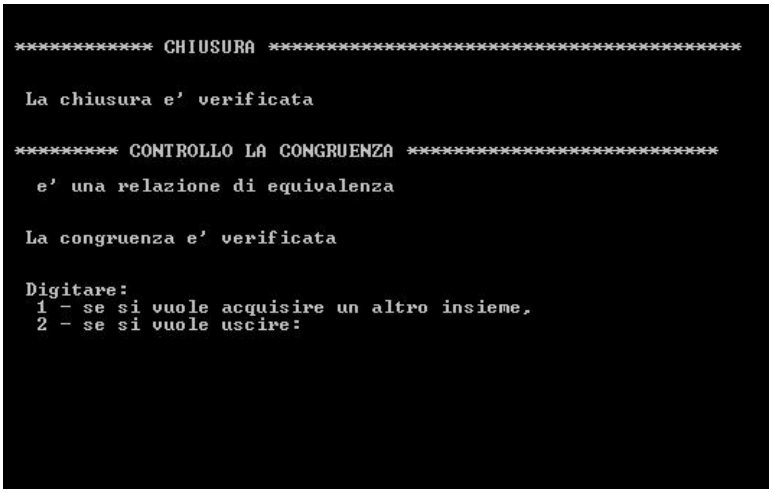
Acquisisco la relazione binaria e la stampo per farla vedere all'utente..

```
********* ACQUISIZIONE DELLA RELAZIONE BINARIA *****************

 Inserisci i termini della coppia
   Primo Termine: 0
   Secondo Termine: 1

 Digitare:
  0 - per terminare l'acquisizione,
  1 - se si vuole acquisire un altra coppia: 1

 Inserisci i termini della coppia
   Primo Termine: 1
   Secondo Termine: 2

 Digitare:
  0 - per terminare l'acquisizione,
  1 - se si vuole acquisire un altra coppia: 0

********* STAMPA DELLA RELAZIONE BINARIA **********************

 La relazione binaria e':

  {(0.00,1.00) ; (1.00,2.00)}
```

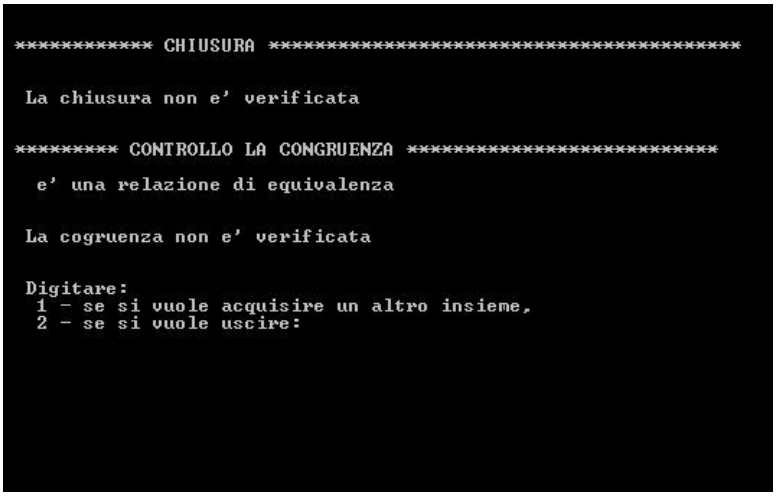
Acquisisco l'operazione acquisendo tutti i risultati possibili..

```
********* ACQUISIZIONE DELL'OPERAZIONE *************************

 Inserire ora i risultati dell'operazioni:

 Digitare 999 per risultati impossibili o indeterminati.

 0.000000 * 0.000000 = 1

 0.000000 * 1.000000 = 2

 0.000000 * 2.000000 = 3

 0.000000 * 3.000000 = 1

 1.000000 * 0.000000 = 2

 1.000000 * 1.000000 = 3

 1.000000 * 2.000000 = 4

 1.000000 * 3.000000 = 5

 2.000000 * 0.000000 = 6

 2.000000 * 1.000000 = 7

 2.000000 * 2.000000 = 8
```

Inserisco un'operazione chiusa rispetto all'insieme e una relazione che sia una congruenza rispetto l'operazione e verifico l'output..



Inserisco un'operazione non chiusa rispetto all'insieme e verifico l'output..



Inserisco un'operazione non chiusa rispetto all'insieme e che la relazione non sia una congruenza rispetto all'operazione e verifico l'output..