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 **1**

2 Analisi del Problema **2**
2.1 Input 2
2.2 Output 2
2.3 Relazioni tra input ed output 2

3 Progettazione dell'algoritmo **3**
3.1 Scelte di progetto 3
3.2 Strutture utilizzate 3
3.3 Passi del programma 3

4 Implementazione dell'algoritmo **4**
4.1 Programma 4
4.2 Makefile 22

5 Testing del programma **23**

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     do
214     {
215         elemento_acquisito = scanf("%d",&zeri);
216         if(elemento_acquisito != 1)
217         {
218             do
219                 carattere_non_letto = getchar();
220             while (carattere_non_letto != '\n');
221         }
222     }
223     while(elemento_acquisito != 1 || (zeri != 1 && zeri != 2));
224     if (zeri == 1)
225     {
226         insieme.elementi_insieme = (double *)
227                                 realloc (insieme.elementi_insieme, (i
                                        +1) * sizeof (double));
228         insieme.elementi_insieme[i] = 0;
229         i = 1;
230     }
231
232     /*faccio partire i i+1 se c'e' lo zero*/
233
234     if(zeri == 2)
235         i = 0;
236
237     printf("\n\n Per terminare l'acquisizione digitare 0\n\n");
238
```

```
239     while(finisci_di_acquisire != 1)
240     {
241         insieme.elementi_insieme = (double *)
242                                 realloc (insieme.elementi_insieme,
243                                          (i+1) * sizeof (double));
244         printf("\n Digitare ora il %d elemento: ",i+1);
245         elemento_acquisito = scanf("%lf",&temporaneo);
246 
247         if(temporaneo == 0)
248         {
249             finisci_di_acquisire = 1;
250             insieme.numero_elementi = i;
251         }
252 
253         if(i >= 0)
254             insieme.elementi_insieme[i] = temporaneo;
255 
256         for(j = i - 1; j >= 0; j--)
257         {
258             if(elemento_acquisito != 1 ||
259                    temporaneo == insieme.elementi_insieme[j])
260             {
261                 do
262                     carattere_non_letto = getchar();
263                 while (carattere_non_letto != '\n');
264                 i--;
265                 j = 0;
266             }
267 
268         }
269         i++;
270     }
271 
272 
273     /***********************/
274     /* stampa dell'insieme */
275     /***********************/
276     printf("\n\n ************** STAMPA DELL'");
277     printf("INSIEME **************************\n");
278     printf("\n\n L'insieme acquisito e':");
279     printf("\n\n { ");
280     i=0;
281     while(i < insieme.numero_elementi)
282     {
283         printf("%.2lf",insieme.elementi_insieme[i]);
284         if(i+1 < insieme.numero_elementi)
285             printf(" ; ");
286         i++;
287     }
```

```
288     printf(" }\n\n");
289
290
291
292     return insieme;
293 }
294
295 insieme_t crea_insieme_vuoto()
296 {
297     insieme_t insieme;
298     insieme.elementi_insieme = (double *) malloc (1);
299     insieme.numero_elementi = 0;
300     return insieme;
301 }
302
303 rel_bin acquisisci_rel_bin(insieme_t insieme)
304 {
305     printf("\n\n ********* ACQUISIZIONE DELLA");
306     printf("RELAZIONE BINARIA ***************\n");
307     rel_bin relazione;
308
309     int acquisizione_finita,
310         risultato_lettura,
311         i,
312         primo_termine_acquisito;
313
314     char carattere_non_letto;
315
316     acquisizione_finita = 1;
317     primo_termine_acquisito = 0;
318
319     relazione.dimensione = 0;
320     relazione.primo_termine = (double *) malloc (2);
321     relazione.secondo_termine = (double *) malloc (2);
322
323     while (acquisizione_finita == 1)
324     {
325         primo_termine_acquisito = 0;
326         relazione.dimensione++;
327         acquisizione_finita = 2;
328
329         /*Acquisisco i termini della coppia*/
330
331         printf ("\n\n Inserisci i termini della coppia \n ");
332         relazione.primo_termine = (double *)
333                                 realloc (relazione.primo_termine,
334                                         (relazione.dimensione+1) *
                                             sizeof (double));
335         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
                    carattere_non_letto = getchar();
                while (carattere_non_letto != '\n');
        }
        while (acquisizione_finita < 0 || acquisizione_finita > 1 );
```

```
383     }
384     return relazione;
385 }
386 
387 /************FUNZIONE DI STAMPA***************/
388 
389 void stampa (rel_bin stampa)
390 {
391 
392     int i = 0;
393     printf("\n\n ******** STAMPA DELLA RELAZIONE BINARIA *****");
394     printf ("****************\n\n La relazione binaria e':");
395     printf ("\n\n {");
396 
397     /******Stampa per coppie numeriche *****/
398 
399     while (i < stampa.dimensione)
400     {
401         printf ("(%.2lf,%.2lf)",
402                 stampa.primo_termine[i],
403                 stampa.secondo_termine[i]);
404         if (i+1 != stampa.dimensione)
405             printf (" ; ");
406         i++;
407     }
408     printf("}\n");
409     return ;
410 }
411 
412 int acquisisci_elemento(insieme_t insieme)
413 {
414     /* dichiaro le variabili */
415     char carattere_non_letto;
416 
417     int lettura_corretta,
418         i,
419         elemento_trovato;
420 
421     double elemento;
422     /* inizializzo le variabili */
423     elemento = 0;
424     lettura_corretta = 1;
425     do
426     {
427         /* controllo che i valori siano stati letti correttamente */
428         /* e nel caso non sia cosi svuoto il buffer */
429         if(lettura_corretta != 1)
430         {
431             do
```

```
432                 carattere_non_letto = getchar();
433             while (carattere_non_letto != '\n');
434             printf ("\n C'e'un errore, reinserire ");
435             printf ("il termine e verificare\n");
436             printf (" che appartenga all'insieme");
437             printf ("precedentemente inserito: \n ");
438         }
439         lettura_corretta = scanf("%lf",&elemento);
440         /* verifico se l'elemento che si vuole utilizzare nella
                relazione */
441         /* e' presente nell'insieme inserito */
442         elemento_trovato = 0;
443         for(i=0; i < insieme.numero_elementi; i++)
444             if(elemento == insieme.elementi_insieme[i])
445                 elemento_trovato = 1;
446
447         if(elemento_trovato == 0)
448             lettura_corretta = 0;
449     }
450     while(lettura_corretta == 0);
451
452     return elemento;
453 }
454
455
456 /* Acquisisco l'operazione*/
457
458 operazione_t acquisisci_operazione(insieme_t insieme)
459 {
460     operazione_t operazione;
461     char carattere_non_letto;
462     int i,
463         j,
464         dimensione,
465         controllo;
466
467     i = 0;
468     j = 0;
469     dimensione = 0;
470
471     operazione.risultati = (double *) malloc (2);
472     operazione.operando_a = (double *) malloc (2);
473     operazione.operando_b = (double *) malloc (2);
474     printf("\n\n ******** ACQUISIZIONE ");
475     printf("DELL'OPERAZIONE **********************\n");
476     printf(" \n\n Inserire ora i risultati dell'operazioni: \n");
477     printf(" \n Digitare 999 per risultati ");
478     printf("impossibili o indeterminati. \n");
479     for(i = 0; i < insieme.numero_elementi; i++)
```

```
480     {
481         for(j = 0; j < insieme.numero_elementi; j++)
482         {
483             operazione.risultati = (double *)
484                                 realloc (operazione.risultati,
485                                         (dimensione+1) * sizeof (
                                               double));
486             operazione.operando_a = (double *)
487                                  realloc (operazione.operando_a,
488                                         (dimensione+1) * sizeof (
                                               double));
489             operazione.operando_b = (double *)
490                                  realloc (operazione.operando_b,
491                                         (dimensione+1) * sizeof (
                                               double));
492             operazione.operando_a[dimensione] = insieme.
                    elementi_insieme[i];
493             operazione.operando_b[dimensione] = insieme.
                    elementi_insieme[j];
494             printf("\n %f * %f = ",insieme.elementi_insieme[i],
495                    insieme.elementi_insieme[j]);
496             do{
497
498                 controllo = scanf("%lf",&operazione.risultati[
                        dimensione]);
499             if(controllo != 1)
500             {
501                 do
502                     carattere_non_letto = getchar();
503                 while (carattere_non_letto != '\n');
504             }
505             }
506
507             while(controllo != 1);
508             dimensione++;
509         }
510     }
511     return operazione;
512 }
513
514 int controllo_chiusura(insieme_t insieme,operazione_t operazione)
515 {
516     int i,
517         j,
518         chiusura;
519
520     i = 0;
521     j = 0;
522     chiusura = 0;
```

```
523
524     for(i = 0; i<(insieme.numero_elementi*insieme.numero_elementi);
            i++)
525     {
526         chiusura = 0;
527         if(operazione.risultati[i] != 999)
528             for(j=0; j<insieme.numero_elementi; j++)
529                 if(operazione.risultati[i] == insieme.elementi_insieme
                        [j])
530                 {
531                     chiusura = 1;
532                     j = insieme.numero_elementi+1;
533                 }
534         if(chiusura == 0)
535             i = (insieme.numero_elementi*insieme.numero_elementi);
536     }
537     printf("\n\n ************ CHIUSURA **********************");
538     printf("******************\n");
539     if(chiusura == 0)
540         printf("\n\n La chiusura non e' verificata\n");
541     if(chiusura == 1)
542         printf("\n\n La chiusura e' verificata\n");
543
544     return chiusura;
545 }
546
547 int controllo_riflessivita (rel_bin verifica)
548 {
549
550     int i,
551         j,
552         k,
553         riscontro,
554         secondo_riscontro,
555         riflessivita;
556
557     riflessivita = 1;
558     i = 0;
559     j = 0;
560     k = 0;
561     riscontro = 0;
562     secondo_riscontro = 0;
563
564     /*Verifica riflessivita'*/
565
566     /*Definizione: una relazione per la quale esiste almeno un
            elemento che non e'in relazione
567     con se' stesso non soddisfa la definizione di riflessivita'*/
568
```

```
569     while ( (i < verifica.dimensione) && (k < verifica.dimensione))
570     {
571
572         /*Verifica riflessivita' per numeri*/
573
574         riscontro = 0;
575         secondo_riscontro = 0;
576         if (verifica.primo_termine[i] == verifica.secondo_termine[i])
577             riscontro++; /**Controllo se c'e' stato un riscontro a,a
                    **/
578         secondo_riscontro++;
579         if (riscontro != 0)
580         {
581             i++;
582             k++;
583         }
584         /**/
585         else
586         {
587             j = 0;
588             riscontro = 0;
589             secondo_riscontro = 0;
590
591             /***************** Controllo la riflessivita' per gli
                    elementi del primo insieme
                    ****************************/
592
593             while (j < verifica.dimensione)
594             {
595                 if (j == i)
596                     j++;
597                 else
598                 {
599                     if (verifica.primo_termine[i] ==
600                             verifica.primo_termine[j])
601                         if (verifica.primo_termine[j] ==
602                                 verifica.secondo_termine[j])
603                             riscontro++;
604
605                     j++;
606                 }
607             }
608
609             j = 0;
610
611             /***************** Controllo la riflessivita' per gli
                    elementi del secondo insieme
                    ****************************/
612
```

```
613             while (j < verifica.dimensione)
614             {
615                 if (j == k)
616                     j++;
617                 else
618                 {
619                     if (verifica.secondo_termine[k] ==
620                             verifica.secondo_termine[j])
621                         if (verifica.primo_termine[j] ==
622                                 verifica.secondo_termine[j])
623                             secondo_riscontro++;
624 
625                     j++;
626                 }
627             }
628             if (riscontro != 0)
629                 i++;
630 
631             /**** Se non c'e' stato un riscontro di riflessivita'
                    esco e imposto la riflessivita' a 0 *****/
632 
633             else
634             {
635                 i = verifica.dimensione;
636                 riflessivita = 0;
637             }
638 
639             if (secondo_riscontro != 0)
640                 k++;
641 
642             else
643             {
644                 k = verifica.dimensione;
645                 riflessivita = 0;
646             }
647         }
648 
649     }
650 
651 
652     /********** Fine riflessivita **********************/
653     return (riflessivita);
654 }
655 
656 int controllo_transitivita (rel_bin verifica)
657 {
658 
659     int i,
660         j,
```

```
661         k,
662         transitivita;
663
664     /*IMPOSTO LA TRANSITIVITA' INIZIALMENTE COME VERA E AZZERO I
            CONTATORI*/
665     transitivita = 1;
666     i = 0;
667     j = 0;
668     k = 0;
669
670     /*VERIFICA TRANSITIVITa' PER NUMERI*/
671
672
673     while (i < verifica.dimensione)
674     {
675         j = 0;
676
677         while (j < verifica.dimensione)
678         {
679             k = 0;
680
681             if (verifica.secondo_termine[i] ==
682                    verifica.primo_termine[j])
683             {
684                 transitivita = 0;
685
686                 while (k < verifica.dimensione)
687                 {
688                     if (verifica.primo_termine[i] ==
689                            verifica.primo_termine[k])
690                     {
691                         if (verifica.secondo_termine[k] ==
692                                verifica.secondo_termine[j])
693                         {
694                             transitivita = 1;
695                             k = verifica.dimensione;
696                         }
697                     }
698
699                     k++;
700                 }
701
702                 if (transitivita==0)
703                 {
704                     j = verifica.dimensione;
705                     i = verifica.dimensione;
706                 }
707             }
708
```

```
709             j++;
710         }
711 
712         i++;
713     }
714 
715     /************ Fine controllo Transitivita' ************/
716 
717     return (transitivita);
718 
719 }
720 
721 
722 int relazione_equivalenza (rel_bin verifica)
723 {
724 
725     int riflessivita,
726         simmetria,
727         transitivita,
728         equivalenza;
729 
730     equivalenza = 0;
731     riflessivita = controllo_riflessivita(verifica);
732     simmetria = controllo_simmetria(verifica);
733     transitivita = controllo_transitivita(verifica);
734 
735     if (riflessivita == 1 && simmetria == 1 && transitivita == 1)
736     {
737         printf ("\n e' una relazione di equivalenza\n");
738         equivalenza=1;
739     }
740 
741     if (riflessivita == 0){
742         printf ("\n non e'una relazione di equivalenza ");
743     printf ("perche' non e' riflessiva\n");
744     }
745     if (simmetria == 0){
746         printf ("\n non e'una relazione di equivalenza ");
747     printf ("perche' non e' simmetrica\n");
748     }
749     if (transitivita == 0){
750         printf ("\n non e'una relazione di equivalenza ");
751     printf ("perche' non e' transitiva\n");
752     }
753     return equivalenza;
754 }
755 
756 int controllo_simmetria (rel_bin verifica)
757 {
```

```
758
759     int i,
760         j,
761         riscontro,
762         simmetria;
763
764     simmetria = 1;
765
766
767     i = 0;
768     j = 0;
769     riscontro = 0;
770
771     /*controllo della simmetria per numeri*/
772
773     while ( i < verifica.dimensione)
774     {
775
776         j = 0;
777         while ( j < verifica.dimensione)
778         {
779
780             if (verifica.primo_termine[i] ==
781                     verifica.secondo_termine[j])
782                 if (verifica.primo_termine[j] ==
783                         verifica.secondo_termine[i])
784                     riscontro++;
785             j++;
786         }
787
788         if (riscontro == 0)
789         {
790             j = verifica.dimensione;
791             i = verifica.dimensione;
792             simmetria = 0;
793         }
794         riscontro = 0;
795         i++;
796     }
797
798     return (simmetria);
799 }
800
801
802 void controllo_congruenza(rel_bin relazione,
803                           insieme_t insieme,
804                           operazione_t operazione,
805                           int chiusura)
806 {
```

```
807     printf("\n\n ********* CONTROLLO LA CONGRUENZA
            **************************\n");
808     int equivalenza,
809         controllo,
810         i,
811         j,
812         k;
813
814     equivalenza = relazione_equivalenza(relazione);
815
816     i = 0;
817     j = 0;
818     k = 0;
819     controllo=1;
820
821     for(i = 0; i<relazione.dimensione; i++)
822     {
823         for(j=0;
824                 j<(insieme.numero_elementi*insieme.numero_elementi);
825                 j++)
826         {
827             if(relazione.primo_termine[i] ==
828                     operazione.operando_a[j])
829                 for(k = 0;
830                         k<(insieme.numero_elementi*insieme.
                                numero_elementi);
831                         k++)
832                     if(relazione.secondo_termine[i] ==
833                             operazione.operando_a[k] &&
834                             operazione.operando_b[j] ==
835                             operazione.operando_b[k])
836
837                         if(operazione.risultati[j]
838                                 != operazione.risultati[k])
839                         {
840                             controllo = 0;
841                             k = (insieme.numero_elementi*insieme.
                                    numero_elementi);
842                             j = (insieme.numero_elementi*insieme.
                                    numero_elementi);
843                             i = relazione.dimensione;
844                         }
845             if(relazione.primo_termine[i] ==
846                     operazione.operando_b[j])
847                 for(k = 0;
848                         k<(insieme.numero_elementi*insieme.
                                numero_elementi);
849                         k++)
850                     if(relazione.secondo_termine[i] ==
```

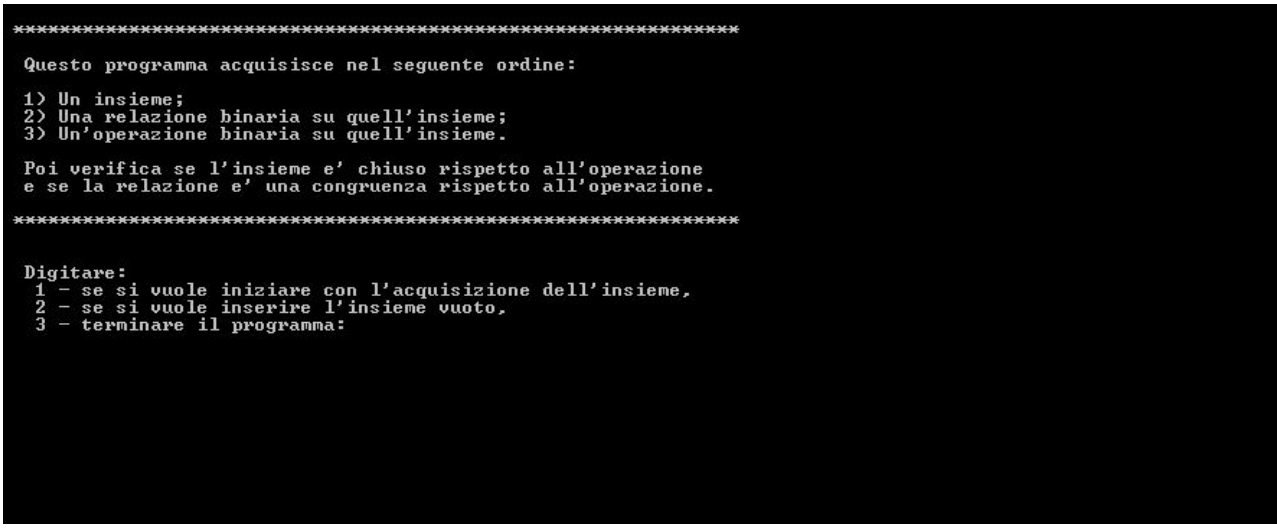
```
851                         operazione.operando_b[k] &&
852                         operazione.operando_a[j] ==
853                         operazione.operando_a[k])
854
855                     if(operazione.risultati[j] !=
856                            operazione.risultati[k])
857                     {
858                         controllo = 0;
859                         k = (insieme.numero_elementi*insieme.
                                numero_elementi);
860                         j = (insieme.numero_elementi*insieme.
                                numero_elementi);
861                         i = relazione.dimensione;
862                     }
863         }
864     }
865
866
867     if(equivalenza == 0 || controllo == 0 || chiusura == 0)
868         printf("\n\n La cogruenza non e' verificata\n");
869     else
870         printf("\n\n La congruenza e' verificata\n");
871
872     return;
873 }
```

4.2 Makefile

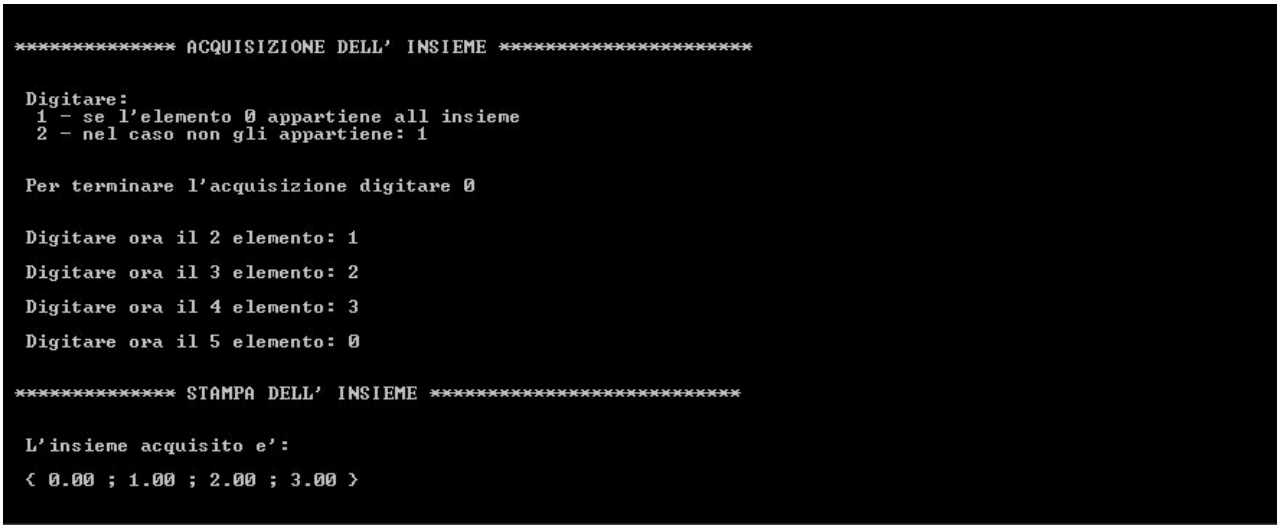
```
1  Progetto_sessione_estiva_PPL: Progetto_sessione_estiva_PPL.c
       Makefile
2     gcc -ansi -Wall -O Progetto_sessione_estiva_PPL.c -o
          Progetto_sessione_estiva_PPL
3  pulisci:
4     rm -f Progetto_sessione_estiva_PPL.o
5  pulisci_tutto:
6     rm -f Progetto_sessione_estiva_PPL Progetto_sessione_estiva_PPL.o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

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
```

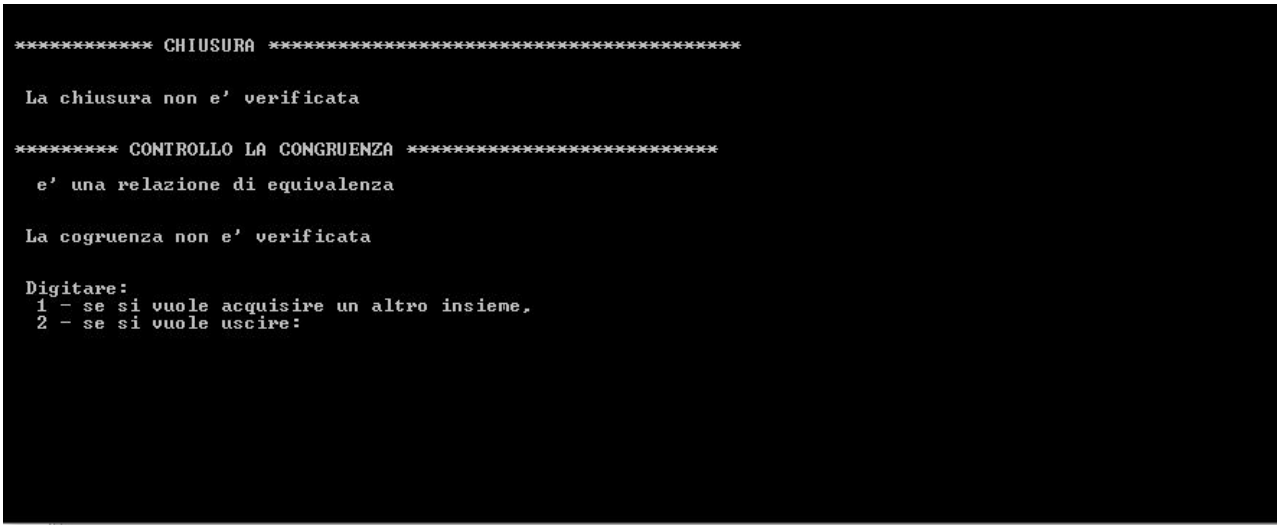
Inserisco un'operazione chiusa rispetto all'insieme e una congruenza e verifico l'output..

```
********** CHIUSURA ****************************************

 La chiusura e' verificata

********* CONTROLLO LA CONGRUENZA ***************************
  e' una relazione di equivalenza

 La congruenza e' verificata

 Digitare:
  1 - se si vuole acquisire un altro insieme,
  2 - se si vuole uscire:
```

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

```
********** CHIUSURA ****************************************

 La chiusura non e' verificata

********* CONTROLLO LA CONGRUENZA ***************************
  e' una relazione di equivalenza

 La cogruenza non e' verificata

 Digitare:
  1 - se si vuole acquisire un altro insieme,
  2 - se si vuole uscire:
```

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


********** CHIUSURA **********
La chiusura non e' verificata
********* CONTROLLO LA CONGRUENZA **********
e' una relazione di equivalenza
La cogruenza non e' verificata
Digitare:
1 - se si vuole acquisire un altro insieme,
2 - se si vuole uscire: