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
17 typedef struct Operazione
18 {
19   double   *operando_a;
20   double   *operando_b;
21   double   *risultati;
22
23 } operazione_t;
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 typedef struct Insieme
36 {
37   double* elementi_insieme;
38   int numero_elementi;
39 } insieme_t;
40
41 /*******************************/
42 /* dichiarazione delle funzioni */
43 /*******************************/
44
```

```
45 int controllo_simmetria (rel_bin);
46 int controllo_riflessivita (rel_bin);
47 int controllo_transitivita (rel_bin);
48 int relazione_equivalenza (rel_bin);
49 insieme_t acquisisci_insieme(void);
50 rel_bin acquisisci_rel_bin(insieme_t);
51 insieme_t crea_insieme_vuoto(void);
52 int acquisisci_elemento(insieme_t);
53 void stampa(rel_bin);
54 operazione_t acquisisci_operazione(insieme_t);
55 int controllo_chiusura(insieme_t,operazione_t);
56 void controllo_congruenza(rel_bin,insieme_t,operazione_t,int);
57
58 /*****************/
59 /* funzione main */
60 /*****************/
61
62 int main()
63 {
64   /*variabile per il controllo dell'acquisizione*/
65   char carattere_non_letto;
66   /*variabile per il controllo della scelta*/
67   int scelta;
68   /*variabile per controllare che la
69   lettura sia avvenuta correttamente*/
70   int lettura_effettuata;
71   /*variabile per dare la possibilita'
72   all'utente di utilizzare il programma
73   piu' di una volta aprendolo solamente
74   una volta*/
75   int ripeti;
76   /*variabile per il salvataggio del
77   risultato della verifica della chiusura*/
78   int chiusura;
79
80   /* variabili per insieme, relazione
81   e operazione*/
82   operazione_t operazione;
83   insieme_t insieme;
84   rel_bin relazione;
85
86   /*inizializzo le variabili*/
87   ripeti = 1;
88   scelta = 0;
89   lettura_effettuata = 0;
90   chiusura = 1;
91
92   while (ripeti == 1)
93   {
```

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

```
143       printf(" sono sempre verificate.\n Per convenzione ");
144       printf("diciamo anche che qualsiasi sia\n l'operazione");
145       printf(" e' chiusa rispetto all'insieme");
146     }
147
148     printf("\n\n Digitare:\n 1 - se si vuole acquisire");
149     printf(" un altro insieme,\n 2 - se si vuole uscire: ");
150
151     do
152     {
153       lettura_effettuata = scanf("%d",&ripeti);
154       if (lettura_effettuata != 1)
155       {
156         do
157           carattere_non_letto = getchar();
158         while (carattere_non_letto != '\n');
159         ripeti = 1;
160       }
161     }
162     while (lettura_effettuata != 1 || (ripeti != 1 && ripeti != 2));
163   }
164
165   return 0;
166 }
167
168
169 /*****************************/
170 /* acquisizione dell'insieme */
171 /*****************************/
172
173 insieme_t acquisisci_insieme()
174 {
175   /*dichiaro la struttura insieme*/
176
177   insieme_t insieme;
178
179   /*variabile contatore */
180   int  i;
181   /*variabile contatore*/
182   int  j;
183   /*variabile per terminare l'acquisizione*/
184   int  finisci_di_acquisire;
185   /*variabile per l'acquisizione dell'elemento 0*/
186   int  zeri;
187   /*variabile per verificare che la
188   acquisizione vada a buon fine*/
189   int  elemento_acquisito;
190   /*variabile necessaria allo
191   svuotamento del buffer*/
```

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

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

```
290   printf("INSIEME **************************\n");
291   printf("\n\n L'insieme acquisito e':");
292   printf("\n\n { ");
293   i=0;
294
295   while (i < insieme.numero_elementi)
296   {
297     printf("%.2lf",insieme.elementi_insieme[i]);
298     if (i+1 < insieme.numero_elementi)
299       printf(" ; ");
300     i++;
301   }
302   printf(" }\n\n");
303
304
305
306   return insieme;
307 }
308
309 insieme_t crea_insieme_vuoto()
310 {
311   /*variabile per la struttura insieme*/
312   insieme_t insieme;
313
314   insieme.elementi_insieme = (double *) malloc (1);
315   insieme.numero_elementi = 0;
316   return insieme;
317 }
318
319 /*Funzione che acquisisce la relazione binaria*/
320
321 rel_bin acquisisci_rel_bin(insieme_t insieme)
322 {
323
324   rel_bin relazione;
325
326   /*variabile utile ad uscire dal ciclo
327   di acquisizione*/
328   int acquisizione_finita,
329   /*variabile per il controllo
330   dell'acquisizione*/
331       risultato_lettura,
332   /*variabile contatore*/
333       i,
334   /*variabile per il primo termine*/
335       primo_termine_acquisito;
336   /*variabile utile in caso si debba
337   svuotare il buffer*/
338   char carattere_non_letto;
```

```
339
340   printf("\n\n ********* ACQUISIZIONE DELLA");
341   printf("RELAZIONE BINARIA ***************\n");
342   /*inizializzo le variabili*/
343   acquisizione_finita = 1;
344   primo_termine_acquisito = 0;
345   relazione.dimensione = 0;
346   /*alloco memoria*/
347   relazione.primo_termine = (double *) malloc (2);
348   relazione.secondo_termine = (double *) malloc (2);
349
350   while (acquisizione_finita == 1)
351   {
352     primo_termine_acquisito = 0;
353     relazione.dimensione++;
354     acquisizione_finita = 2;
355
356     /*Acquisisco i termini della coppia*/
357
358     printf ("\n\n Inserisci i termini della coppia \n ");
359
360     relazione.primo_termine = (double *)
361                             realloc (relazione.primo_termine,
362                                     (relazione.dimensione+1)
363                                     * sizeof (double));
364
365     relazione.secondo_termine = (double *)
366                               realloc (relazione.secondo_termine,
367                                       (relazione.dimensione+1)
368                                       * sizeof (double));
369     risultato_lettura = 0;
370
371
372     /*Acquisisco il primo termine*/
373     if (primo_termine_acquisito == 0)
374     {
375       printf (" Primo Termine: ");
376       relazione.primo_termine[relazione.dimensione - 1] =
377         acquisisci_elemento(insieme);
378     }
379     primo_termine_acquisito = 1;
380
381     /*Acquisisco il secondo termine*/
382     if (primo_termine_acquisito == 1)
383     {
384       printf (" Secondo Termine: ");
385       relazione.secondo_termine[relazione.dimensione - 1]
386         = acquisisci_elemento(insieme);
387
```

```
388       for (i=relazione.dimensione-2; i>=0; i--)
389         if (relazione.primo_termine[relazione.dimensione - 1]
390             == relazione.primo_termine[i])
391           if (relazione.secondo_termine[relazione.dimensione -1]
392               == relazione.secondo_termine[i])
393           {
394             relazione.dimensione--;
395             i = 0;
396           }
397     }
398
399     /*Chiedo all'utente se ci sono altre coppie*/
400
401     do
402     {
403       printf("\n\n Digitare:\n 0 - per");
404       printf("terminare l'acquisizione,");
405       printf("\n 1 - se si vuole acquisire un altra coppia: ");
406       risultato_lettura = scanf ("%d",
407                                  &acquisizione_finita);
408       if (acquisizione_finita < 0 ||
409           acquisizione_finita > 1 || risultato_lettura != 1)
410         do
411           carattere_non_letto = getchar();
412         while (carattere_non_letto != '\n');
413     }
414     while (acquisizione_finita < 0 || acquisizione_finita > 1 );
415   }
416
417   return relazione;
418 }
419
420 /************FUNZIONE DI STAMPA***************/
421
422 void stampa (rel_bin stampa)
423 {
424   /*variabile contatore*/
425   int i = 0;
426
427   printf("\n\n ********* STAMPA DELLA RELAZIONE BINARIA *****");
428   printf ("****************\n\n La relazione binaria e':");
429   printf ("\n\n {");
430
431   /******Stampa per coppie numeriche *****/
432
433   while (i < stampa.dimensione)
434   {
435     printf ("(%.2lf,%.2lf)",
436             stampa.primo_termine[i],
```

```
437             stampa.secondo_termine[i]);
438     if (i+1 != stampa.dimensione)
439       printf (" ; ");
440     i++;
441   }
442   printf("}\n");
443   return;
444 }
445
446 int acquisisci_elemento(insieme_t insieme)
447 {
448   /* variabile necessaria per il controllo
449   dell'acquisizione*/
450   char carattere_non_letto;
451   /*variabile per il controllare che
452   gli elementi acquisiti siano stati
453   letti correttamente*/
454   int lettura_corretta,
455   /*variabile contatore*/
456       i,
457   /*variabile di controllo per verificare
458   la non ripetizione di elementi*/
459       elemento_trovato;
460
461   double elemento;
462   /* inizializzo le variabili */
463   elemento = 0;
464   lettura_corretta = 1;
465
466   do
467   {
468     /* controllo che i valori siano
469         stati letti correttamente
470      e nel caso svuoto il buffer */
471
472     if (lettura_corretta != 1)
473     {
474       do
475         carattere_non_letto = getchar();
476       while (carattere_non_letto != '\n');
477       printf ("\n C'e'un errore, reinserire ");
478       printf ("il termine e verificare\n");
479       printf (" che appartenga all'insieme");
480       printf ("precedentemente inserito: \n ");
481     }
482     lettura_corretta = scanf("%lf",&elemento);
483
484     /* verifico se l'elemento che si
485        vuole utilizzare nella relazione
```

```
486       e' presente nell'insieme inserito */
487
488     elemento_trovato = 0;
489
490     for (i=0; i < insieme.numero_elementi; i++)
491       if (elemento == insieme.elementi_insieme[i])
492         elemento_trovato = 1;
493
494     if (elemento_trovato == 0)
495       lettura_corretta = 0;
496   }
497   while (lettura_corretta == 0);
498
499   return elemento;
500 }
501
502
503 /* Acquisisco l'operazione*/
504
505 operazione_t acquisisci_operazione(insieme_t insieme)
506 {
507     /*variabile per acquisire l operazione*/
508   operazione_t operazione;
509     /*variabile per svuotare il buffer*/
510   char carattere_non_letto;
511     /*variabile contatore*/
512   int i,
513       j,
514     /*variabile per settare la dimensione dell'array*/
515       dimensione,
516     /*variabile per il controllo*/
517       controllo;
518
519   i = 0;
520   j = 0;
521   dimensione = 0;
522
523   operazione.risultati = (double *) malloc (2);
524   operazione.operando_a = (double *) malloc (2);
525   operazione.operando_b = (double *) malloc (2);
526   printf("\n\n ********* ACQUISIZIONE ");
527   printf("DELL'OPERAZIONE **********************\n");
528   printf(" \n\n Inserire ora i risultati dell'operazioni: \n");
529   printf(" \n Digitare 999 per risultati ");
530   printf("impossibili o indeterminati. \n");
531
532   for (i = 0; i < insieme.numero_elementi; i++)
533   {
534     for (j = 0; j < insieme.numero_elementi; j++)
```

```
535     {
536       operazione.risultati = (double *)
537                           realloc (operazione.risultati,
538                                    (dimensione+1)
539                                    * sizeof (double));
540       operazione.operando_a = (double *)
541                            realloc (operazione.operando_a,
542                                     (dimensione+1)
543                                     * sizeof (double));
544       operazione.operando_b = (double *)
545                            realloc (operazione.operando_b,
546                                     (dimensione+1)
547                                     * sizeof (double));
548       operazione.operando_a[dimensione] = insieme.elementi_insieme[i
              ];
549       operazione.operando_b[dimensione] = insieme.elementi_insieme[j
              ];
550       printf("\n %f * %f = ",insieme.elementi_insieme[i],
551              insieme.elementi_insieme[j]);
552       do
553       {
554
555         controllo = scanf("%lf",
556                           &operazione.risultati[dimensione]);
557         if (controllo != 1)
558         {
559           do
560             carattere_non_letto = getchar();
561           while (carattere_non_letto != '\n');
562         }
563       }
564
565       while (controllo != 1);
566       dimensione++;
567     }
568   }
569   return operazione;
570 }
571
572 int controllo_chiusura(insieme_t insieme,operazione_t operazione)
573 {
574   int i,
575       j,
576       chiusura;
577
578   i = 0;
579   j = 0;
580   chiusura = 0;
581
```

```
582   for (i = 0;
583       i<(insieme.numero_elementi*insieme.numero_elementi);
584       i++)
585   {
586     chiusura = 0;
587     if (operazione.risultati[i] != 999)
588       for (j=0; j<insieme.numero_elementi; j++)
589         if (operazione.risultati[i] ==
590             insieme.elementi_insieme[j])
591         {
592           chiusura = 1;
593           j = insieme.numero_elementi+1;
594         }
595     if (chiusura == 0)
596       i = (insieme.numero_elementi*insieme.numero_elementi);
597   }
598   printf("\n\n ************ CHIUSURA **********************");
599   printf("******************\n");
600   if (chiusura == 0)
601     printf("\n\n La chiusura non e' verificata\n");
602   if (chiusura == 1)
603     printf("\n\n La chiusura e' verificata\n");
604
605   return chiusura;
606 }
607
608 int controllo_riflessivita (rel_bin verifica)
609 {
610     /*variabile contatore*/
611   int i,
612       j,
613       k,
614     /*variabili per contare i riscontri*/
615       riscontro,
616       secondo_riscontro,
617     /*variabile per vedere se e' stata verificata la riflessivita'*/
618       riflessivita;
619
620   riflessivita = 1;
621   i = 0;
622   j = 0;
623   k = 0;
624   riscontro = 0;
625   secondo_riscontro = 0;
626
627   /*Verifica riflessivita'*/
628
629   /*Definizione: una relazione per la quale
630    esiste almeno un elemento che non e'in relazione
```

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

```
680       {
681         if (j == k)
682           j++;
683         else
684         {
685           if (verifica.secondo_termine[k] ==
686               verifica.secondo_termine[j])
687             if (verifica.primo_termine[j] ==
688                 verifica.secondo_termine[j])
689               secondo_riscontro++;
690
691           j++;
692         }
693       }
694       if (riscontro != 0)
695         i++;
696
697       /**** Se non c'e' stato un riscontro di riflessivita'
698        esco e imposto la riflessivita' a 0 *****/
699
700       else
701       {
702         i = verifica.dimensione;
703         riflessivita = 0;
704       }
705
706       if (secondo_riscontro != 0)
707         k++;
708
709       else
710       {
711         k = verifica.dimensione;
712         riflessivita = 0;
713       }
714     }
715
716   }
717
718
719   /********** Fine riflessivita **********************/
720   return (riflessivita);
721 }
722
723 int controllo_transitivita (rel_bin verifica)
724 {
725     /*variabile contatore*/
726   int i,
727       j,
728       k,
```

```
729     /*variabile per controllare la transitivita'*/
730       transitivita;
731
732   /*IMPOSTO LA TRANSITIVITA' INIZIALMENTE COME VERA
733    E AZZERO I CONTATORI*/
734
735   transitivita = 1;
736   i = 0;
737   j = 0;
738   k = 0;
739
740   /*VERIFICA TRANSITIVITa' PER NUMERI*/
741
742
743   while (i < verifica.dimensione)
744   {
745     j = 0;
746
747     while (j < verifica.dimensione)
748     {
749       k = 0;
750
751       if (verifica.secondo_termine[i] ==
752           verifica.primo_termine[j])
753       {
754         transitivita = 0;
755
756         while (k < verifica.dimensione)
757         {
758           if (verifica.primo_termine[i] ==
759               verifica.primo_termine[k])
760           {
761             if (verifica.secondo_termine[k] ==
762                 verifica.secondo_termine[j])
763             {
764               transitivita = 1;
765               k = verifica.dimensione;
766             }
767           }
768
769           k++;
770         }
771
772         if (transitivita==0)
773         {
774           j = verifica.dimensione;
775           i = verifica.dimensione;
776         }
777       }
```

```
778
779       j++;
780     }
781
782     i++;
783   }
784
785   /************ Fine controllo Transitivita' ************/
786
787   return (transitivita);
788
789 }
790
791
792 int relazione_equivalenza (rel_bin verifica)
793 {
794
795 /*variabili per controllare le propieta' dell'equivalenza*/
796   int riflessivita,
797       simmetria,
798       transitivita,
799       equivalenza;
800
801   equivalenza = 0;
802   riflessivita = controllo_riflessivita(verifica);
803   simmetria = controllo_simmetria(verifica);
804   transitivita = controllo_transitivita(verifica);
805
806   if (riflessivita == 1 && simmetria == 1 && transitivita == 1)
807   {
808     printf ("\n e' una relazione di equivalenza\n");
809     equivalenza=1;
810   }
811
812   if (riflessivita == 0)
813   {
814     printf ("\n non e'una relazione di equivalenza ");
815     printf ("perche' non e' riflessiva\n");
816   }
817   if (simmetria == 0)
818   {
819     printf ("\n non e'una relazione di equivalenza ");
820     printf ("perche' non e' simmetrica\n");
821   }
822   if (transitivita == 0)
823   {
824     printf ("\n non e'una relazione di equivalenza ");
825     printf ("perche' non e' transitiva\n");
826   }
```

```
827   return equivalenza;
828 }
829
830 int controllo_simmetria (rel_bin verifica)
831 {
832     /*variabili contatore*/
833   int i,
834       j,
835     /*variabile per controllare se c'e' stato un riscontro*/
836       riscontro,
837     /*variabile per controllare la simmetria*/
838       simmetria;
839
840   simmetria = 1;
841
842
843   i = 0;
844   j = 0;
845   riscontro = 0;
846
847   /*controllo della simmetria per numeri*/
848
849   while ( i < verifica.dimensione)
850   {
851
852     j = 0;
853     while ( j < verifica.dimensione)
854     {
855
856       if (verifica.primo_termine[i] ==
857           verifica.secondo_termine[j])
858         if (verifica.primo_termine[j] ==
859             verifica.secondo_termine[i])
860           riscontro++;
861       j++;
862     }
863
864     if (riscontro == 0)
865     {
866       j = verifica.dimensione;
867       i = verifica.dimensione;
868       simmetria = 0;
869     }
870     riscontro = 0;
871     i++;
872   }
873
874   return (simmetria);
875 }
```

```
876
877
878 void controllo_congruenza(rel_bin relazione,
879                           insieme_t insieme,
880                           operazione_t operazione,
881                           int chiusura)
882 {
883   printf("\n\n ********* CONTROLLO LA CONGRUENZA");
884   printf(" ***************************\n");
885   /*variabile per il controllo dell'equivalenza*/
886   int equivalenza,
887    /*variabile di controllo*/
888       controllo,
889     /*variabili contatori*/
890       i,
891       j,
892       k;
893
894   equivalenza = relazione_equivalenza(relazione);
895
896   i = 0;
897   j = 0;
898   k = 0;
899   controllo=1;
900
901   for (i = 0; i<relazione.dimensione; i++)
902   {
903     for (j=0;
904         j<(insieme.numero_elementi*insieme.numero_elementi);
905         j++)
906     {
907       if (relazione.primo_termine[i] ==
908           operazione.operando_a[j])
909         for (k = 0;
910             k<(insieme.numero_elementi*insieme.numero_elementi);
911             k++)
912           if (relazione.secondo_termine[i] ==
913               operazione.operando_a[k] &&
914               operazione.operando_b[j] ==
915               operazione.operando_b[k])
916
917             if (operazione.risultati[j]
918                 != operazione.risultati[k])
919             {
920               controllo = 0;
921               k = (insieme.numero_elementi*
922                    insieme.numero_elementi);
923
924               j = (insieme.numero_elementi*
```

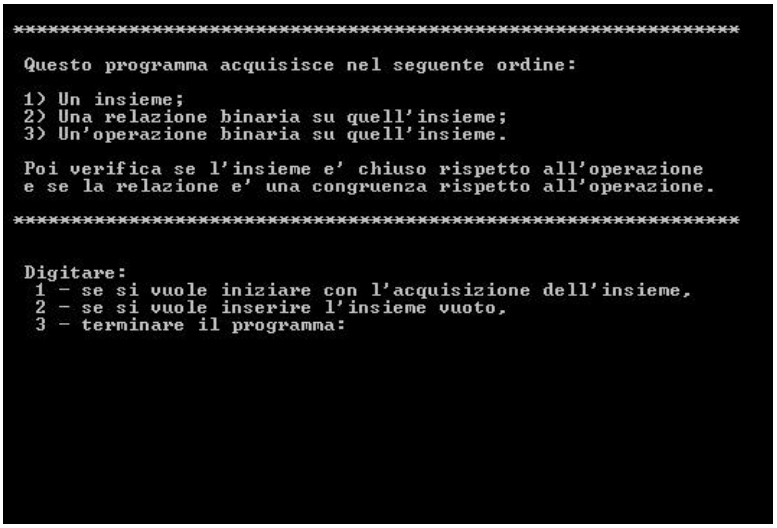
```
                insieme.numero_elementi);

            i = relazione.dimensione;
          }
      if (relazione.primo_termine[i] ==
          operazione.operando_b[j])
        for (k = 0;
          k < insieme.numero_elementi*insieme.numero_elementi;
          k++)
          if (relazione.secondo_termine[i] ==
            operazione.operando_b[k] &&
            operazione.operando_a[j] ==
            operazione.operando_a[k])

            if (operazione.risultati[j] !=
              operazione.risultati[k])
            {
              controllo = 0;
              k = insieme.numero_elementi*
                  insieme.numero_elementi;

              j = insieme.numero_elementi*
                  insieme.numero_elementi;

              i = relazione.dimensione;
            }
    }
  }


  if (equivalenza == 0 || controllo == 0 || chiusura == 0)
    printf("\n\n La cogruenza non e' verificata\n");
  else
    printf("\n\n La congruenza e' verificata\n");

  return;
}
```

4.2 Makefile

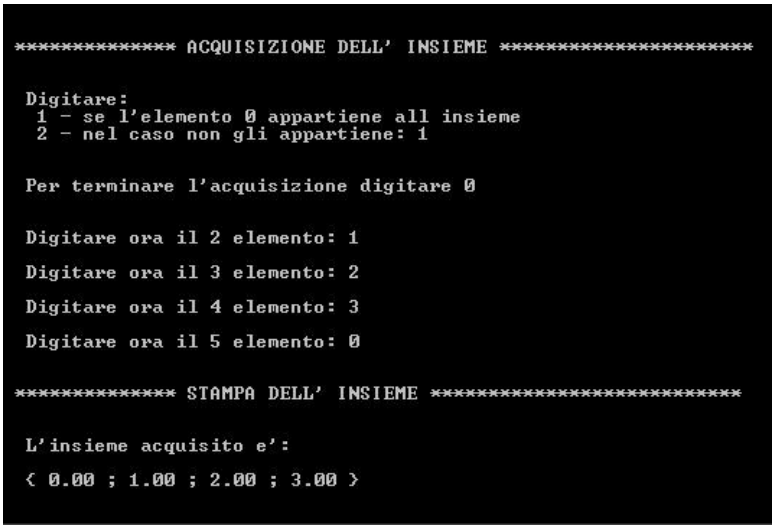
```
1 Progetto_sessione_estiva_PPL: Progetto_sessione_estiva_PPL.c
      Makefile
2   gcc -ansi -Wall -O Progetto_sessione_estiva_PPL.c -o
        Progetto_sessione_estiva_PPL
3 pulisci:
4   rm -f Progetto_sessione_estiva_PPL.o
5 pulisci_tutto:
6   rm -f Progetto_sessione_estiva_PPL Progetto_sessione_estiva_PPL.o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

5 Testing del programma

Spiego all'utente cosa fa il programma..

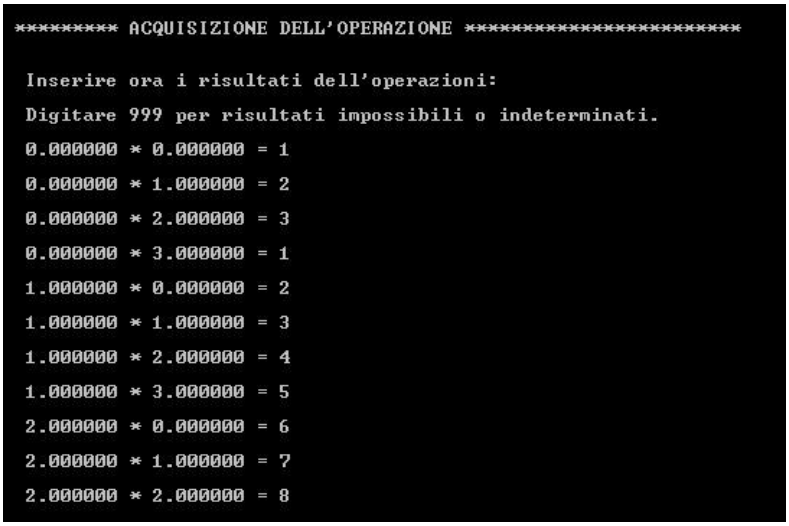


Acquisisco l'insieme e lo stampo per farlo vedere all'utente..

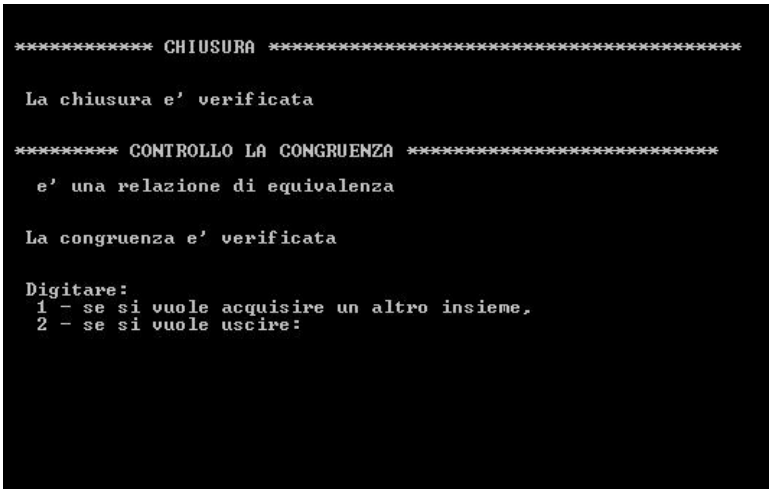


Acquisisco la relazione binaria e la stampo per farla vedere all'utente..

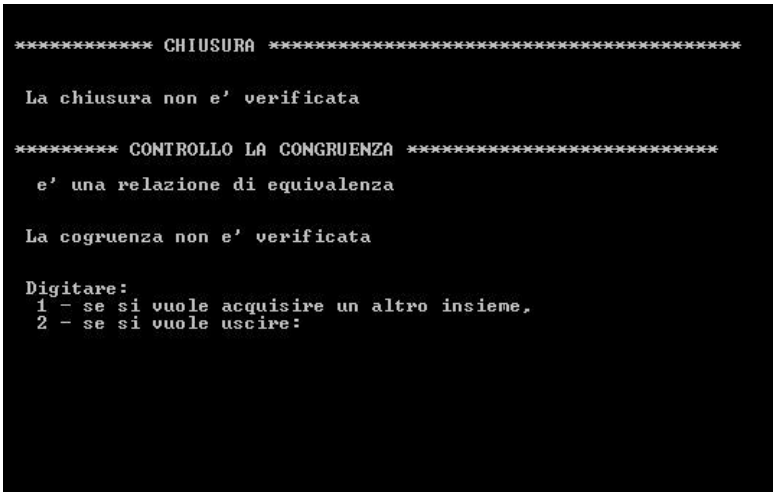
Acquisisco l'operazione acquisendo tutti i risultati possibili..



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..