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 24

5 Testing del programma **25**

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 - nel caso non gli appartiene: ");
222
223   do
224   {
225     elemento_acquisito = scanf("%d",&zeri);
226     if (elemento_acquisito != 1)
227     {
228       do
229         carattere_non_letto = getchar();
230       while (carattere_non_letto != '\n');
231     }
232   }
233   while (elemento_acquisito != 1 || (zeri != 1 && zeri != 2));
234
235   if (zeri == 1)
236   {
237     insieme.elementi_insieme = (double *)
238                              realloc (insieme.elementi_insieme,
239                                       (i+1) * sizeof (double));
240     insieme.elementi_insieme[i] = 0;
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

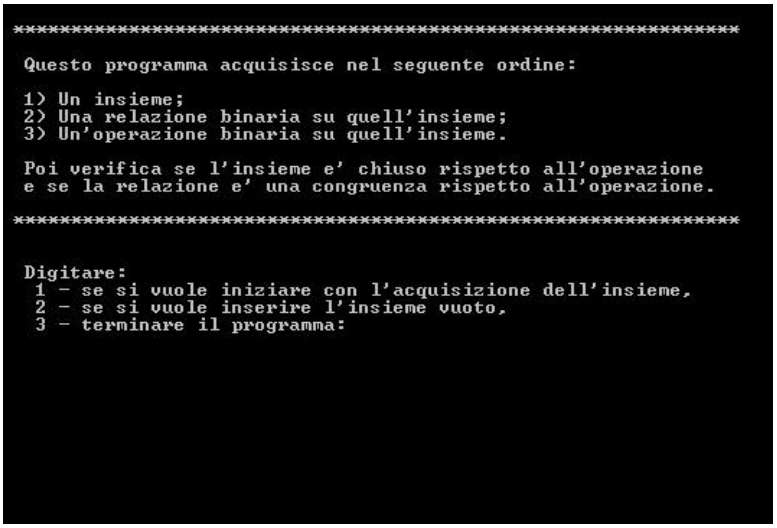
```
925
926             i = relazione.dimensione;
927           }
928       if (relazione.primo_termine[i] ==
929          operazione.operando_b[j])
930         for (k = 0;
931           k < insieme.numero_elementi*insieme.numero_elementi;
932           k++)
933         if (relazione.secondo_termine[i] ==
934             operazione.operando_b[k] &&
935             operazione.operando_a[j] ==
936             operazione.operando_a[k])
937
938           if (operazione.risultati[j] !=
939               operazione.risultati[k])
940           {
941             controllo = 0;
942             k = insieme.numero_elementi*
943                 insieme.numero_elementi;
944
945             j = insieme.numero_elementi*
946                 insieme.numero_elementi;
947
948             i = relazione.dimensione;
949           }
950     }
951   }
952
953
954   if (equivalenza == 0 || controllo == 0 || chiusura == 0)
955     printf("\n\n La cogruenza non e' verificata\n");
956   else
957     printf("\n\n La congruenza e' verificata\n");
958
959   return;
960 }
```

4.2 Makefile

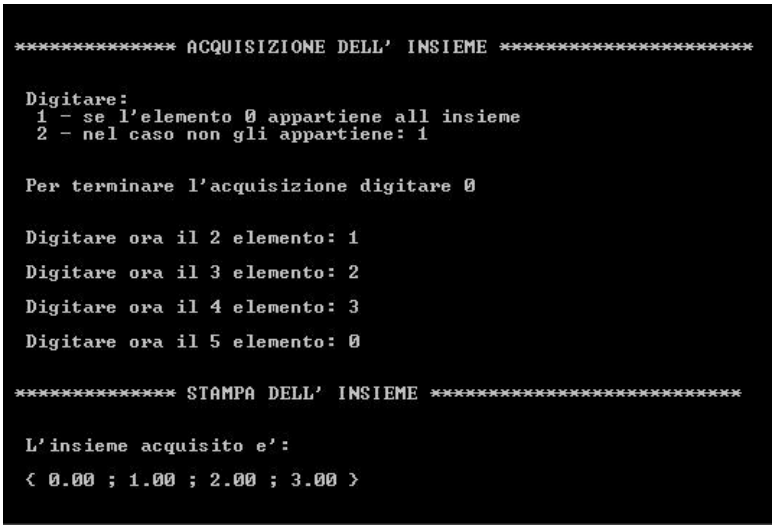
```
1 Progetto_sessione_estiva_PPL: Progetto_sessione_estiva_PPL.c
      Makefile
2   gcc -ansi -Wall -O Progetto_sessione_estiva_PPL.c -o
        Progetto_sessione_estiva_PPL
3 pulisci:
4   rm -f Progetto_sessione_estiva_PPL.o
5 pulisci_tutto:
6   rm -f Progetto_sessione_estiva_PPL Progetto_sessione_estiva_PPL.o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

5 Testing del programma

Spiego all'utente cosa fa il programma..

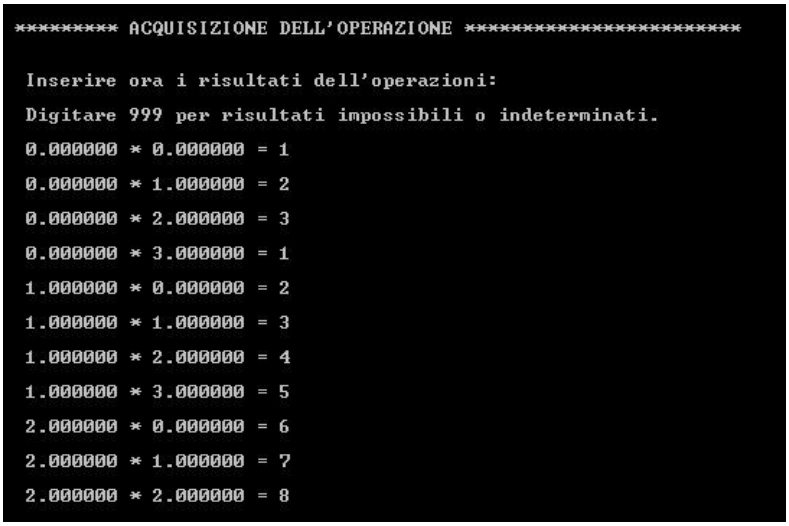


Acquisisco l'insieme e lo stampo per farlo vedere all'utente..

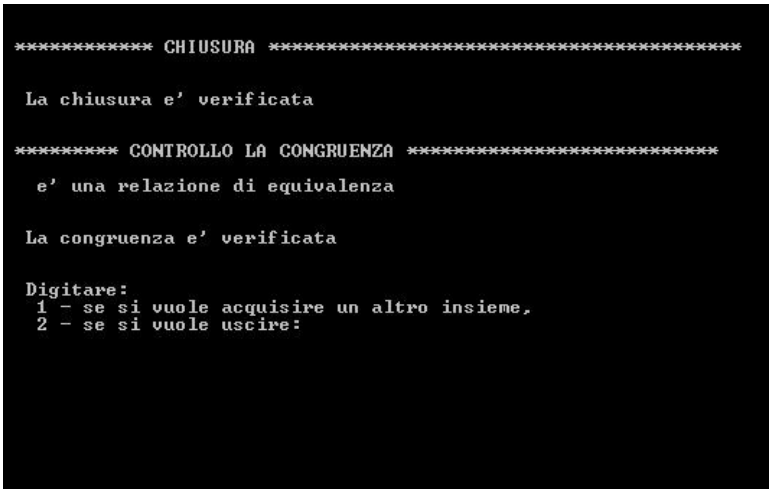


Acquisisco la relazione binaria e la stampo per farla vedere all'utente..

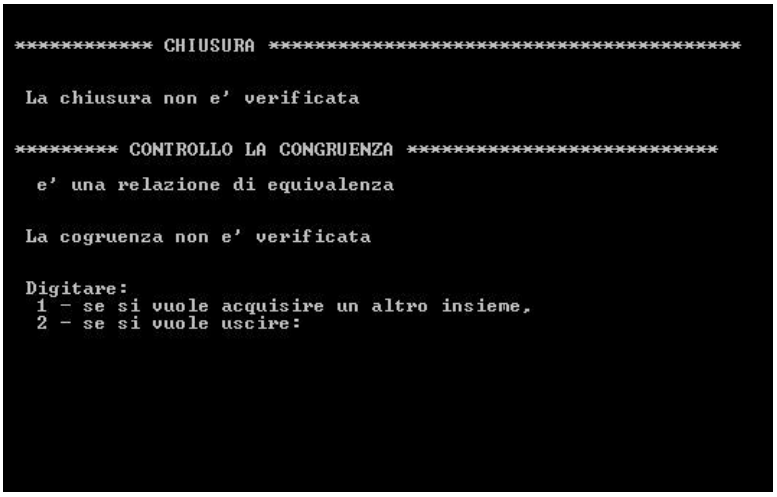
Acquisisco l'operazione acquisendo tutti i risultati possibili..



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..