

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

June 28, 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	26
5	Testing del programma	27
6	Verifica del programma	33

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.

Per l'insieme vuoto do la possibilità all'utente sia di scegliere a priori che l'insieme deve essere vuoto sia di poterlo immettere durante la creazione dell'insieme.

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  
25 typedef struct RelBin  
26 {  
27     /* coppia numerica */  
28  
29     double    *primo_termine;  
30     double    *secondo_termine;  
31  
32     /* variabile per sapere il numero delle coppie */  
33  
34     int dimensione;  
35 } rel_bin;  
36  
37 typedef struct Insieme  
38 {  
39     double* elementi_insieme;  
40     int numero_elementi;  
41 } insieme_t;  
42  
43 /*****  
44 /* dichiarazione delle funzioni */
```

```

45  /*****/
46
47  void errore(void);
48  void svuota_buffer(void);
49  insieme_t acquisisci_insieme(void);
50  void stampa_insieme(insieme_t);
51  insieme_t crea_insieme_vuoto(void);
52  rel_bin acquisisci_rel_bin(insieme_t);
53  int acquisisci_elemento(insieme_t);
54  void stampa_rel_bin(rel_bin);
55  operazione_t acquisisci_operazione(insieme_t);
56  int controllo_simmetria(rel_bin);
57  int controllo_riflessivita(rel_bin);
58  int controllo_transitivita(rel_bin);
59  int relazione_equivalenza(rel_bin);
60  int controllo_chiusura(insieme_t,
61                        operazione_t);
62  void controllo_congruenza(rel_bin,
63                           insieme_t,
64                           operazione_t,
65                           int);
66  void informazioni_sul_programma(void);
67  void scelta_operazione(void);
68  int ripeti(void);
69
70  /*****/
71  /* funzione main */
72  /*****/
73
74  int main()
75  {
76      /*variabile per dare la possibilita'
77      all'utente di utilizzare il programma
78      ricorsivamente*/
79      int ripetere;
80      ripetere = 1;
81
82      informazioni_sul_programma();
83
84      while (ripetere == 1)
85      {
86
87          scelta_operazione();
88          ripetere = ripeti();
89      }
90
91
92      return 0;
93  }

```

```

94
95 /*funzione per poter riportare un segnale di errore dopo un
    acquisizione da tastiera errata*/
96 void errore()
97 {
98     printf("\n\n hai inserito un valore errato");
99     printf("\n inserire un valore corretto: ");
100
101     return;
102 }
103
104 /*funzione per poter pulire il buffer*/
105 void svuota_buffer()
106 {
107     /*variabile per svuotare il buffer*/
108     char carattere_non_letto;
109     do
110     {
111         carattere_non_letto = getchar();
112     }
113     while (carattere_non_letto != '\n');
114
115     return;
116 }
117
118 /*informazioni sul programma*/
119 void informazioni_sul_programma()
120 {
121     printf("\n *****");
122     printf("*****\n");
123     printf("\n Questo programma acquisisce nel seguente");
124     printf(" ordine:\n");
125     printf("\n 1) Un insieme;\n 2) Una relazione binaria su ");
126     printf("quell'insieme;\n 3) Un'operazione binaria su quell");
127     printf("'insieme.\n\n Poi verifica se l'insieme e' chiuso ");
128     printf("rispetto all'operazione \n ");
129     printf(" e se la relazione e' una");
130     printf(" congruenza rispetto all'operazione.\n");
131
132     return;
133 }
134
135 /*Funzione per chiedere all'utente cosa vuole fare*/
136 void scelta_operazione()
137 {
138     /*variabile per il controllo della scelta*/
139     int scelta;
140     /*variabile per controllare che la
141     lettura sia avvenuta correttamente*/

```



```

142 int lettura_effettuata;
143 /*variabile per il salvataggio del
144 risultato della verifica della chiusura*/
145 int chiusura;
146
147 /* variabili per insieme, relazione
148 e operazione*/
149 operazione_t operazione;
150 insieme_t insieme;
151 rel_bin relazione;
152
153 /*inizializzo le variabili*/
154 scelta = 0;
155 lettura_effettuata = 0;
156 chiusura = 1;
157
158 printf("\n *****");
159 printf("*****\n");
160 printf("\n\n Digitare:\n 1 - se si vuole iniziare con");
161 printf(" 1'acquisizione dell'insieme,\n 2 - se si vuole ");
162 printf("inserire l'insieme vuoto,");
163 printf("\n 3 - se si vogliono avere ");
164 printf("informazioni sul programma ");
165 printf("\n 4 - uscire da questo menu': ");
166
167 do
168 {
169     lettura_effettuata = scanf("%d",&scelta);
170     if (lettura_effettuata != 1)
171     {
172         errore();
173         svuota_buffer();
174         scelta=4;
175     }
176 }
177 while ((scelta != 1 && scelta != 2
178         && scelta != 3 && scelta != 4) || lettura_effettuata != 1);
179
180 if (scelta == 1)
181 {
182
183     insieme = acquisisci_insieme();
184     stampa_insieme(insieme);
185     if (insieme.numero_elementi != 0)
186     {
187         relazione = acquisisci_rel_bin(insieme);
188         stampa_rel_bin(relazione);
189         operazione = acquisisci_operazione(insieme);
190         chiusura = controllo_chiusura(insieme,

```

```

191         operazione);
192     controllo_congruenza(relazione,
193         insieme,
194         operazione,
195         chiusura);
196 }
197 }
198 if (scelta == 2 || insieme.numero_elementi == 0)
199 {
200
201     printf("\n\n ***** INSIEME");
202     printf(" VUOTO *****\n");
203     insieme = crea_insieme_vuoto();
204     printf("\n L'insieme che si e' scelto e' vuoto,");
205     printf(" quindi qualsiasi \n sia la relazione");
206     printf(", simmetria, riflessivita' e transitivita'\n");
207     printf(" sono sempre verificate.\n Per convenzione ");
208     printf("diciamo anche che qualsiasi sia\n l'operazione");
209     printf(" e' chiusa rispetto all'insieme");
210 }
211
212 if (scelta == 3)
213     informazioni_sul_programma();
214
215 return;
216 }
217
218
219 /*Funzione per ripetere il procedimento*/
220 int ripeti()
221 {
222     int ripetere;
223     int lettura_effettuata;
224     printf("\n\n Digitare:\n 1 - se si vuole eseguire");
225     printf(" un'altra operazione,");
226     printf("\n 2 - se si vuole terminare il programma: ");
227     do
228     {
229         lettura_effettuata = scanf("%d",&ripetere);
230         if (lettura_effettuata != 1 )
231         {
232             errore();
233             svuota_buffer();
234             ripetere = 1;
235         }
236     }
237     while (lettura_effettuata != 1 || (ripetere != 1 && ripetere != 2)
238         );

```

```

239
240     return (ripetere);
241 }
242
243
244 /*****
245  * acquisizione dell'insieme */
246 *****/
247
248 insieme_t acquisisci_insieme()
249 {
250     /*dichiaro la struttura insieme*/
251
252     insieme_t insieme;
253
254     /*variabile contatore */
255     int i;
256     /*variabile per il controllo di doppioni*/
257     int buffer_vuoto;
258     /*variabile contatore*/
259     int j;
260     /*variabile per controllare la fine dell acquisizione*/
261     int controllo;
262     /*variabile per terminare l'acquisizione*/
263     int finisci_di_acquisire;
264     /*variabile per verificare che la
265     acquisizione vada a buon fine*/
266     int elemento_acquisito;
267     /*variabile necessaria allo
268     svuotamento del buffer*/
269     char carattere_non_letto;
270     /*variabile per acquisire ogni
271     elemento temporaneamente*/
272     double temporaneo;
273
274     /*inizializzo le variabili*/
275     elemento_acquisito = 0;
276     j = 0;
277     i = 0;
278     temporaneo = 0;
279     insieme.numero_elementi = 50;
280     finisci_di_acquisire = 0;
281     controllo = 0;
282     buffer_vuoto = 1;
283     /*alloco memoria*/
284     insieme.elementi_insieme = (double *)
285                               malloc (insieme.numero_elementi);
286
287     /*inizio la vera e propria acquisizione*/

```

```

288
289 printf("\n\n Si e' scelto di acquisire un'insieme\n");
290
291 /*chiedo se l'utente vuole inserire lo 0*/
292
293 printf("\n\n ***** ACQUISIZIONE DELL'");
294 printf("INSIEME *****\n");
295
296 printf("\n\n Per terminare l'acquisizione digitare a\n\n");
297
298 while (finisci_di_acquisire != 1)
299 {
300     controllo = 0;
301     insieme.elementi_insieme = (double *)
302         realloc (insieme.elementi_insieme,
303             (i+1) * sizeof (double));
304     printf("\n Digitare ora il %d elemento: ",i+1);
305     /*svuoto il buffer prima di acquisire*/
306     if (buffer_vuoto != 1)
307         svuota_buffer();
308
309     buffer_vuoto = 0;
310     elemento_acquisito = scanf("%lf",&temporaneo);
311     if (i >= 0 && finisci_di_acquisire != 1 )
312         insieme.elementi_insieme[i] = temporaneo;
313
314     /*controllo se c'e' stato un errore nell'acquisizione*/
315     if (elemento_acquisito != 1)
316     {
317         do
318         {
319             carattere_non_letto = getchar();
320             /*controllo se l'utente ha digitato il carattere di
321                terminazione*/
322             if ((carattere_non_letto == 'a') && (controllo == 0))
323             {
324                 finisci_di_acquisire = 1;
325                 insieme.numero_elementi = i;
326             }
327             /*controllo che mi permette di verificare se e' stato
328                digitato solo un carattere */
329             if (controllo > 1)
330                 finisci_di_acquisire = 0;
331             controllo++;
332         }
333         while (carattere_non_letto != '\n');
334     }
335     /*mi segno che il buffer e' gia' vuoto per non andarlo a
336        svuotare una seconda volta*/

```

```

334     buffer_vuoto = 1;
335     i--;
336 }
337
338 /*controllo che l'elemento non sia gia presente nell insieme*/
339
340 for (j = i-1; j >= 0; j--)
341 {
342     if (temporaneo == insieme.elementi_insieme[j])
343     {
344         i--;
345         j = 0;
346     }
347 }
348 i++;
349 }
350
351 return (insieme);
352 }
353
354 /*****/
355 /* stampa dell'insieme */
356 /*****/
357
358 void stampa_insieme(insieme_t insieme)
359 {
360     /*variabile contatore*/
361     int i;
362
363     printf("\n\n ***** STAMPA DELL'");
364     printf("INSIEME *****\n");
365     printf("\n\n L'insieme acquisito e':");
366     printf("\n\n { ");
367     i=0;
368
369     while (i < insieme.numero_elementi)
370     {
371         printf("%.2lf",insieme.elementi_insieme[i]);
372         if (i+1 < insieme.numero_elementi)
373             printf(" ; ");
374         i++;
375     }
376     printf(" }\n\n");
377
378     return;
379 }
380
381 insieme_t crea_insieme_vuoto()
382 {

```

```

383 /*variabile per la struttura insieme*/
384 insieme_t insieme;
385
386 insieme.elementi_insieme = (double *) malloc (1);
387 insieme.numero_elementi = 0;
388 return (insieme);
389 }
390
391 /*Funzione che acquisisce la relazione binaria*/
392
393 rel_bin acquisisci_rel_bin(insieme_t insieme)
394 {
395
396     rel_bin relazione;
397
398     /*variabile utile ad uscire dal ciclo
399     di acquisizione*/
400     int acquisizione_finita,
401         /*variabile per il controllo
402         dell'acquisizione*/
403         risultato_lettura,
404         /*variabile contatore*/
405         i,
406         /*relazione vuota*/
407         relazione_vuota,
408         /*variabile per il primo termine*/
409         primo_termine_acquisito;
410
411     printf("\n\n ***** ACQUISIZIONE DELLA");
412     printf("RELAZIONE BINARIA *****\n");
413     printf("\n\n Si vuole acquisire una relazione vuota?");
414     printf("\n\n Digitare:\n 0 - si\n");
415     printf(" 1 - no: ");
416     do
417     {
418         risultato_lettura = scanf("%d",&relazione_vuota);
419         if (risultato_lettura != 1 || relazione_vuota != 0 &&
420             relazione_vuota != 1 )
421         {
422             errore();
423             svuota_buffer();
424         }
425     } while (risultato_lettura != 1 || relazione_vuota != 0 &&
426             relazione_vuota != 1);
427     if (relazione_vuota == 0)
428         printf(" si e' scelto di inserire una relazione vuota");
429
430     /*inizializzo le variabili*/

```

```

430     acquisizione_finita = 1;
431     primo_termine_acquisito = 0;
432     relazione.dimensione = 0;
433     /*alloca memoria*/
434     relazione.primo_termine = (double *) malloc (2);
435     relazione.secondo_termine = (double *) malloc (2);
436     if (relazione_vuota == 1)
437     {
438         while (acquisizione_finita == 1)
439         {
440             primo_termine_acquisito = 0;
441             relazione.dimensione++;
442             acquisizione_finita = 2;
443
444             /*Acquisisco i termini della coppia*/
445
446             printf ("\n\n Inserisci i termini della coppia \n ");
447
448             relazione.primo_termine = (double *)
449                                     realloc (relazione.primo_termine,
450                                               (relazione.dimensione+1)
451                                               * sizeof (double));
452
453             relazione.secondo_termine = (double *)
454                                         realloc (relazione.secondo_termine,
455                                                   (relazione.dimensione+1)
456                                                   * sizeof (double));
457             risultato_lettura = 0;
458
459
460             /*Acquisisco il primo termine*/
461             if (primo_termine_acquisito == 0)
462             {
463                 printf (" Primo Termine: ");
464                 relazione.primo_termine[relazione.dimensione - 1] =
465                     acquisisci_elemento(insieme);
466             }
467             primo_termine_acquisito = 1;
468
469             /*Acquisisco il secondo termine*/
470             if (primo_termine_acquisito == 1)
471             {
472                 printf (" Secondo Termine: ");
473                 relazione.secondo_termine[relazione.dimensione - 1]
474                     = acquisisci_elemento(insieme);
475
476                 for (i=relazione.dimensione-2; i>=0; i--)
477                     if (relazione.primo_termine[relazione.dimensione - 1]
478                         == relazione.primo_termine[i])

```

```

479         if (relazione.secondo_termine[relazione.dimensione -1]
480             == relazione.secondo_termine[i])
481         {
482             relazione.dimensione--;
483             i = 0;
484         }
485     }
486
487     /*Chiedo all'utente se ci sono altre coppie*/
488
489     do
490     {
491         printf("\n\n Digitare:\n 0 - per");
492         printf(" terminare l'acquisizione,");
493         printf("\n 1 - se si vuole acquisire un'altra coppia: ");
494         risultato_lettura = scanf ("%d",
495                                     &acquisizione_finita);
496         if (acquisizione_finita < 0 ||
497             acquisizione_finita > 1 || risultato_lettura != 1)
498         {
499             errore();
500             svuota_buffer();
501         }
502     }
503     while (acquisizione_finita < 0 || acquisizione_finita > 1 );
504 }
505 }
506 svuota_buffer();
507 return (relazione);
508 }
509
510 /*****FUNZIONE DI STAMPA*****/
511
512 void stampa_rel_bin(rel_bin stampa)
513 {
514     /*variabile contatore*/
515     int i = 0;
516
517     printf("\n\n ***** STAMPA DELLA RELAZIONE BINARIA *****");
518     printf ("*****\n\n La relazione binaria e':");
519     printf ("\n\n {");
520
521     /*****Stampa per coppie numeriche *****/
522
523     while (i < stampa.dimensione)
524     {
525         printf ("(%.2lf,%.2lf)",
526                 stampa.primo_termine[i],
527                 stampa.secondo_termine[i]);

```



```

528     if (i+1 != stampa.dimensione)
529         printf (" ; ");
530     i++;
531 }
532 printf("}\n");
533 return;
534 }
535
536 int acquisisci_elemento(insieme_t insieme)
537 {
538     /*variabile per il controllare che
539     gli elementi acquisiti siano stati
540     letti correttamente*/
541     int lettura_corretta,
542         /*variabile contatore*/
543         i,
544         /*variabile di controllo per verificare
545         la non ripetizione di elementi*/
546         elemento_trovato;
547
548     double elemento;
549     /* inizializzo le variabili */
550     elemento = 0;
551     lettura_corretta = 1;
552
553     do
554     {
555         /* controllo che i valori siano
556         stati letti correttamente
557         e nel caso svuoto il buffer */
558
559         if (lettura_corretta != 1)
560         {
561             svuota_buffer();
562             printf ("\n verificare che l'elemento");
563             printf (" appartenga \n all'insieme");
564             printf (" precedentemente inserito. \n ");
565             errore();
566         }
567         lettura_corretta = scanf("%lf",&elemento);
568
569         /* verifico se l'elemento che si
570         vuole utilizzare nella relazione
571         e' presente nell'insieme inserito */
572
573         elemento_trovato = 0;
574
575         for (i=0; i < insieme.numero_elementi; i++)
576             if (elemento == insieme.elementi_insieme[i])

```

```

577     elemento_trovato = 1;
578
579     if (elemento_trovato == 0)
580         lettura_corretta = 0;
581 }
582 while (lettura_corretta == 0);
583
584 svuota_buffer();
585 return (elemento);
586 }
587
588
589 /* Acquisisco l'operazione*/
590
591 operazione_t acquisisci_operazione(insieme_t insieme)
592 {
593     /*variabile per acquisire l operazione*/
594     operazione_t operazione;
595     /*variabile contatore*/
596     int i,
597         j,
598         /*variabile che stabilisce la dimensione dell'array*/
599         dimensione,
600         /*variabile per il controllo*/
601         controllo;
602
603     i = 0;
604     j = 0;
605     dimensione = 0;
606
607     operazione.risultati = (double *) malloc (2);
608     operazione.operando_a = (double *) malloc (2);
609     operazione.operando_b = (double *) malloc (2);
610     printf("\n\n ***** ACQUISIZIONE ");
611     printf("DELL'OPERAZIONE *****\n");
612     printf(" \n\n Inserire ora i risultati dell'operazioni: \n");
613     printf(" \n Digitare 999 per risultati ");
614     printf("impossibili o indeterminati. \n");
615
616     for (i = 0; i < insieme.numero_elementi; i++)
617     {
618         for (j = 0; j < insieme.numero_elementi; j++)
619         {
620             operazione.risultati = (double *)
621                                     realloc (operazione.risultati,
622                                               (dimensione+1)
623                                               * sizeof (double));
624             operazione.operando_a = (double *)
625                                     realloc (operazione.operando_a,

```

```

626                                     (dimensione+1)
627                                     * sizeof (double));
628     operazione.operando_b = (double *)
629                             realloc (operazione.operando_b,
630                                     (dimensione+1)
631                                     * sizeof (double));
632     operazione.operando_a[dimensione] = insieme.elementi_insieme[i
        ];
633     operazione.operando_b[dimensione] = insieme.elementi_insieme[j
        ];
634     printf("\n %f * %f = ", insieme.elementi_insieme[i],
635           insieme.elementi_insieme[j]);
636     do
637     {
638
639         controllo = scanf("%lf",
640                           &operazione.risultati[dimensione]);
641         if (controllo != 1)
642         {
643             errore();
644             svuota_buffer();
645         }
646     }
647
648     while (controllo != 1);
649     dimensione++;
650 }
651 }
652 svuota_buffer();
653 return (operazione);
654 }
655
656 int controllo_chiusura(insieme_t insieme,
657                       operazione_t operazione)
658 {
659     int i,
660         j,
661         chiusura;
662
663     i = 0;
664     j = 0;
665     chiusura = 0;
666
667     for (i = 0;
668         i < (insieme.numero_elementi * insieme.numero_elementi);
669         i++)
670     {
671         chiusura = 0;
672         if (operazione.risultati[i] != 999)

```

```

673     for (j=0; j<insieme.numero_elementi; j++)
674         if (operazione.risultati[i] ==
675             insieme.elementi_insieme[j])
676         {
677             chiusura = 1;
678             j = insieme.numero_elementi+1;
679         }
680     if (chiusura == 0)
681         i = (insieme.numero_elementi*insieme.numero_elementi);
682 }
683 printf("\n\n ***** CHIUSURA *****");
684 printf("*****\n");
685 if (chiusura == 0)
686     printf("\n\n La chiusura non e' verificata\n");
687 if (chiusura == 1)
688     printf("\n\n La chiusura e' verificata\n");
689
690 return (chiusura);
691 }
692
693 int controllo_riflessivita(rel_bin verifica)
694 {
695     /*variabile contatore*/
696     int i,
697         j,
698         k,
699     /*variabili per contare i riscontri*/
700     riscontro,
701     secondo_riscontro,
702     /*variabile per vedere se e' stata verificata la riflessivita
703     '*/
704     riflessivita;
705
706     riflessivita = 1;
707     i = 0;
708     j = 0;
709     k = 0;
710     riscontro = 0;
711     secondo_riscontro = 0;
712
713     /*Verifica riflessivita'*/
714
715     /*Definizione: una relazione per la quale
716     esiste almeno un elemento che non e' in relazione
717     con se' stesso non soddisfa la definizione di riflessivita'*/
718
719     while ( (i < verifica.dimensione) && (k < verifica.dimensione))
720     {

```

```

721 /*Verifica riflessivita' per numeri*/
722
723 riscontro = 0;
724 secondo_riscontro = 0;
725 if (verifica.primo_termine[i] == verifica.secondo_termine[i])
726     riscontro++;/*Controllo se c'e' stato un riscontro a,a*/
727 secondo_riscontro++;
728 if (riscontro != 0)
729 {
730     i++;
731     k++;
732 }
733 /**/
734 else
735 {
736     j = 0;
737     riscontro = 0;
738     secondo_riscontro = 0;
739
740 /* Controllo la riflessivita' per
741 gli elementi del primo insieme */
742
743 while (j < verifica.dimensione)
744 {
745     if (j == i)
746         j++;
747     else
748     {
749         if (verifica.primo_termine[i] ==
750             verifica.primo_termine[j])
751             if (verifica.primo_termine[j] ==
752                 verifica.secondo_termine[j])
753                 riscontro++;
754
755         j++;
756     }
757 }
758
759 j = 0;
760
761 /*Controllo la riflessivita' per gli
762 elementi del secondo insieme*/
763
764 while (j < verifica.dimensione)
765 {
766     if (j == k)
767         j++;
768     else
769     {

```

```

770         if (verifica.secondo_termine[k] ==
771             verifica.secondo_termine[j])
772             if (verifica.primo_termine[j] ==
773                 verifica.secondo_termine[j])
774                 secondo_riscontro++;
775
776         j++;
777     }
778 }
779 if (riscontro != 0)
780     i++;
781
782 /**** Se non c'e' stato un riscontro di riflessivita'
783 esco e imposto la riflessivita' a 0 *****/
784
785 else
786 {
787     i = verifica.dimensione;
788     riflessivita = 0;
789 }
790
791 if (secondo_riscontro != 0)
792     k++;
793
794 else
795 {
796     k = verifica.dimensione;
797     riflessivita = 0;
798 }
799 }
800
801 }
802
803
804 /***** Fine riflessivita *****/
805 return (riflessivita);
806 }
807
808 int controllo_transitivita(rel_bin verifica)
809 {
810     /*variabile contatore*/
811     int i,
812         j,
813         k,
814         /*variabile per controllare la transitivita'*/
815         transitivita;
816
817     /*IMPOSTO LA TRANSITIVITA' INIZIALMENTE COME VERA
818     E AZZERO I CONTATORI*/

```

```

819
820     transitivita = 1;
821     i = 0;
822     j = 0;
823     k = 0;
824
825     /*VERIFICA TRANSITIVITA' PER NUMERI*/
826
827
828     while (i < verifica.dimensione)
829     {
830         j = 0;
831
832         while (j < verifica.dimensione)
833         {
834             k = 0;
835
836             if (verifica.secondo_termine[i] ==
837                 verifica.primo_termine[j])
838             {
839                 transitivita = 0;
840
841                 while (k < verifica.dimensione)
842                 {
843                     if (verifica.primo_termine[i] ==
844                         verifica.primo_termine[k])
845                     {
846                         if (verifica.secondo_termine[k] ==
847                             verifica.secondo_termine[j])
848                         {
849                             transitivita = 1;
850                             k = verifica.dimensione;
851                         }
852                     }
853
854                     k++;
855                 }
856
857                 if (transitivita==0)
858                 {
859                     j = verifica.dimensione;
860                     i = verifica.dimensione;
861                 }
862             }
863
864             j++;
865         }
866
867         i++;

```

```

868 }
869
870 /***** Fine controllo Transitivita' *****/
871
872 return (transitivita);
873
874 }
875
876 int controllo_simmetria(rel_bin verifica)
877 {
878     /*variabili contatore*/
879     int i,
880         j,
881         /*variabile per controllare se c'e' stato un riscontro*/
882         riscontro,
883         /*variabile per controllare la simmetria*/
884         simmetria;
885
886     simmetria = 1;
887
888
889     i = 0;
890     j = 0;
891     riscontro = 0;
892
893     /*controllo della simmetria per numeri*/
894
895     while ( i < verifica.dimensione)
896     {
897
898         j = 0;
899         while ( j < verifica.dimensione)
900         {
901
902             if (verifica.primo_termine[i] ==
903                 verifica.secondo_termine[j])
904                 if (verifica.primo_termine[j] ==
905                     verifica.secondo_termine[i])
906                     riscontro++;
907             j++;
908         }
909
910         if (riscontro == 0)
911         {
912             j = verifica.dimensione;
913             i = verifica.dimensione;
914             simmetria = 0;
915         }
916         riscontro = 0;

```



```

917     i++;
918 }
919
920     return (simmetria);
921 }
922
923
924 int relazione_equivalenza(rel_bin verifica)
925 {
926     /*variabili per controllare le proprieta' dell'equivalenza*/
927     int riflessivita,
928         simmetria,
929         transitivita,
930         equivalenza;
931
932
933     equivalenza = 0;
934     riflessivita = controllo_riflessivita(verifica);
935     simmetria = controllo_simmetria(verifica);
936     transitivita = controllo_transitivita(verifica);
937
938     if (riflessivita == 1 && simmetria == 1 && transitivita == 1)
939     {
940         printf ("\n E' una relazione di equivalenza\n");
941         equivalenza = 1;
942     }
943
944     if (riflessivita == 0)
945     {
946         printf ("\n Non e'una relazione di equivalenza ");
947         printf ("perche' non e' riflessiva\n");
948     }
949     if (simmetria == 0)
950     {
951         printf ("\n Non e'una relazione di equivalenza ");
952         printf ("perche' non e' simmetrica\n");
953     }
954     if (transitivita == 0)
955     {
956         printf ("\n Non e'una relazione di equivalenza ");
957         printf ("perche' non e' transitiva\n");
958     }
959     return (equivalenza);
960 }
961
962
963 void controllo_congruenza(rel_bin relazione,
964                             insieme_t insieme,
965                             operazione_t operazione,

```

```

966             int chiusura)
967 {
968     printf("\n\n ***** CONTROLLO LA CONGRUENZA");
969     printf(" *****\n");
970     /*variabile per il controllo dell'equivalenza*/
971     int equivalenza,
972         /*variabile di controllo*/
973         controllo,
974         /*variabili contatori*/
975         i,
976         j,
977         k;
978     if (relazione.dimensione != 0)
979         equivalenza = relazione_equivalenza(relazione);
980     else
981     {
982         printf("\n\n La relazione vuota ");
983         printf("non e' una relazione di equivalenza");
984         equivalenza = 0;
985     }
986     i = 0;
987     j = 0;
988     k = 0;
989     controllo = 1;
990
991     for (i = 0; i < relazione.dimensione; i++)
992     {
993         for (j = 0;
994             j < (insieme.numero_elementi * insieme.numero_elementi);
995             j++)
996         {
997             if (relazione.primo_termine[i] ==
998                 operazione.operando_a[j])
999                 for (k = 0;
1000                     k < (insieme.numero_elementi * insieme.numero_elementi);
1001                     k++)
1002                     if (relazione.secondo_termine[i] ==
1003                         operazione.operando_a[k] &&
1004                         operazione.operando_b[j] ==
1005                         operazione.operando_b[k])
1006
1007                         if (operazione.risultati[j]
1008                             != operazione.risultati[k])
1009                         {
1010                             controllo = 0;
1011                             k = (insieme.numero_elementi *
1012                                 insieme.numero_elementi);
1013
1014                             j = (insieme.numero_elementi *

```

```

1015         insieme.numero_elementi);
1016
1017         i = relazione.dimensione;
1018     }
1019     if (relazione.primo_termine[i] ==
1020         operazione.operando_b[j])
1021     for (k = 0;
1022         k < insieme.numero_elementi*insieme.numero_elementi;
1023         k++)
1024     if (relazione.secondo_termine[i] ==
1025         operazione.operando_b[k] &&
1026         operazione.operando_a[j] ==
1027         operazione.operando_a[k])
1028
1029         if (operazione.risultati[j] !=
1030             operazione.risultati[k])
1031         {
1032             controllo = 0;
1033             k = insieme.numero_elementi*
1034                 insieme.numero_elementi;
1035
1036             j = insieme.numero_elementi*
1037                 insieme.numero_elementi;
1038
1039             i = relazione.dimensione;
1040         }
1041     }
1042 }
1043
1044
1045 if (equivalenza == 0 || controllo == 0 || chiusura == 0)
1046     printf("\n\n La congruenza non e' verificata\n");
1047 else
1048     printf("\n\n La congruenza e' verificata\n");
1049
1050 return;
1051 }

```

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.

```
*****
Questo programma acquisisce nel seguente ordine:
1> Un insieme;
2> Una relazione binaria su quell'insieme;
3> Un'operazione binaria su quell'insieme.

Poi verifica se l'insieme e' chiuso rispetto all'operazione
e se la relazione e' una congruenza rispetto all'operazione.
*****

Digitare:
1 - se si vuole iniziare con l'acquisizione dell'insieme,
2 - se si vuole inserire l'insieme vuoto,
3 - se si vogliono avere informazioni sul programma
4 - terminare il programma:
```

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

```
***** ACQUISIZIONE DELL' INSIEME *****

Digitare:
1 - se l'elemento 0 appartiene all insieme
2 - nel caso non gli appartiene: 1

Per terminare l'acquisizione digitare 0

Digitare ora il 2 elemento: 1
Digitare ora il 3 elemento: 2
Digitare ora il 4 elemento: 3
Digitare ora il 5 elemento: 0

***** STAMPA DELL' INSIEME *****

L'insieme acquisito e':
< 0.00 ; 1.00 ; 2.00 ; 3.00 >
```

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

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

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

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

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

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

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

La relazione binaria e':

<(0.00,1.00) ; (1.00,2.00)>
```

Acquisisco l'operazione acquisendo tutti i risultati possibili.

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

Inserire ora i risultati dell'operazioni:
Digitare 999 per risultati impossibili o indeterminati.

0.000000 * 0.000000 = 1
0.000000 * 1.000000 = 2
0.000000 * 2.000000 = 3
0.000000 * 3.000000 = 1
1.000000 * 0.000000 = 2
1.000000 * 1.000000 = 3
1.000000 * 2.000000 = 4
1.000000 * 3.000000 = 5
2.000000 * 0.000000 = 6
2.000000 * 1.000000 = 7
2.000000 * 2.000000 = 8
```

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

```
***** 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 congruenza non e' verificata  
  
Digitare:  
1 - se si vuole eseguire un'altra operazione,  
2 - se si vuole uscire:
```

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

```
***** CHIUSURA *****  
  
La chiusura non e' verificata  
  
***** CONTROLLO LA CONGRUENZA *****  
  
Non e'una relazione di equivalenza perche' non e' riflessiva  
Non e'una relazione di equivalenza perche' non e' simmetrica  
  
La congruenza non e' verificata  
  
Digitare:  
1 - se si vuole eseguire un'altra operazione,  
2 - se si vuole uscire:
```

Verifico la presenza di errori durante l'acquisizione della scelta iniziale.

```
*****  
  
Questo programma acquisisce nel seguente ordine:  
  
1) Un insieme;  
2) Una relazione binaria su quell'insieme;  
3) Un'operazione binaria su quell'insieme.  
  
Poi verifica se l'insieme e' chiuso rispetto all'operazione  
e se la relazione e' una congruenza rispetto all'operazione.  
  
*****  
  
Digitare:  
1 - se si vuole iniziare con l'acquisizione dell'insieme,  
2 - se si vuole inserire l'insieme vuoto,  
3 - terminare il programma: a  
  
hai inserito un valore sbagliato  
reinserire: c  
  
hai inserito un valore sbagliato  
reinserire: d  
  
hai inserito un valore sbagliato  
reinserire:
```


Verifico la presenza di errori durante l'acquisizione dell'insieme.

```
Si e' scelto di acquisire un'insieme

***** ACQUISIZIONE DELL'INSIEME *****

Per terminare l'acquisizione digitare a

Digitare ora il 1 elemento: w
Digitare ora il 1 elemento: d
Digitare ora il 1 elemento: e
Digitare ora il 1 elemento: r
Digitare ora il 1 elemento: f
Digitare ora il 1 elemento: g
Digitare ora il 1 elemento: t
Digitare ora il 1 elemento: b
Digitare ora il 1 elemento: d
Digitare ora il 1 elemento:
```

Verifico la presenza di errori durante l'acquisizione della relazione.

```
L'insieme acquisito e':
< 1.00 >

***** ACQUISIZIONE DELLARELAZIONE BINARIA *****

Si vuole acquisire una relazione vuota?
Digitare:
0 - si
1 - no: 1

Inserisci i termini della coppia
Primo Termine: 2
verificare che l'elemento appartenga
all'insieme precedentemente inserito.

hai inserito un valore sbagliato
reinserire: 3
verificare che l'elemento appartenga
all'insieme precedentemente inserito.
```

Acquisisco l'insieme vuoto.

```
***** ACQUISIZIONE DELL'INSIEME *****

Per terminare l'acquisizione digitare a

Digitare ora il 1 elemento: a

***** STAMPA DELL'INSIEME *****

L'insieme acquisito e':
< >

***** INSIEME VUOTO *****

L'insieme che si e' scelto e' vuoto, quindi qualsiasi
sia la relazione, simmetria, riflessivita' e transitivita'
sono sempre verificate.
Per convenzione diciamo anche che qualsiasi sia
l'operazione e' chiusa rispetto all'insieme

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

6 Verifica del programma

Di seguito viene verificata la correttezza di una parte del programma attraverso l'utilizzo di una tripla di Hoare. Si dice tripla di Hoare una tripla nella seguente forma: $\{Q\} S \{R\}$

Dove Q è un predicato detto preconditione, S è un'istruzione ed R è un predicato detto postcondizione. La tripla è vera se e solo se l'esecuzione dell'istruzione S inizia in uno stato della computazione in cui Q è soddisfatta, e termina raggiungendo uno stato della computazione in cui R è soddisfatta.

Istruzione selezionata:

$i=0;$

if (riscontro $\neq 0$)

{

$i++;$

}

sia $\text{riscontro} \neq 0 \vee \text{riscontro} == 0$

$\text{wp}(S,R) = ((\beta \rightarrow \text{wp}(S,R)) \wedge ((\neg\beta) \rightarrow \text{wp}(S,R)))$

$R \Rightarrow (i = 0) \vee (i = 1)$

$P \Rightarrow ((\text{riscontro} \neq 0) \rightarrow ((i = 1) \vee (i = 0)))$

$((\text{riscontro} == 0) \rightarrow ((i = 1) \vee (i = 0)))$

ovvero:

$((\text{riscontro} \neq 0) \rightarrow ((i = 1) \vee (i = 0))) = \text{Vero}$

$((\text{riscontro} == 0) \rightarrow ((i = 1) \vee (i = 0))) = \text{Vero}$

$(\text{Vero} \wedge \text{Vero}) = \text{Vero}$