

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

January 24, 2015

Contents

1	Specifica del Problema	1
2	Analisi del Problema	2
2.1	Input	2
2.2	Output	2
3	Progettazione dell' Algoritmo	3
3.1	Teoria	3
3.2	Funzioni per l'acquisizione:	5
3.3	Funzioni per la verifica delle proprietà:	5
3.4	Funzioni principali:	6
3.5	Input	7
3.6	Output - Acquisizione	8
3.7	Output - stampa	8
3.8	Output - ordine_parziale	8
3.9	Output - ordine_totale	8
3.10	Output - relazione_equivalenza	9
3.11	Output - check_funzione	9
4	Implementazione dell' algoritmo	10
4.1	Libreria	10
4.2	Test	11
4.3	Makefile	13
5	Testing del programma	14
5.1	Test 1:	14
5.2	Test 2:	15
5.3	Test 3:	16
5.4	Test 4:	17
5.5	Test 5:	18
5.6	Test 6:	19
5.7	Test 7:	20
5.8	Test 8:	21
5.9	Test 9:	22
5.10	Test 10:	23
6	Verica del programma	24

1 Specifica del Problema

Write an ANSI C library that manages binary relations by exporting the following functions. The first C function returns a binary relation introduced through the keyboard. The second C function has a binary relation as input parameter and prints it to the screen. The third C function has a binary relation as input parameter and establishes whether it is a partial order relation, printing to the screen which property does not hold in the case that the relation is not such. The fourth C function has a binary relation as input parameter and establishes whether it is a total order relation, printing to the screen which property does not hold in the case that the relation is not such. The fifth C function has a binary relation as input parameter and establishes whether it is an equivalence relation, printing to the screen which property does not hold in the case that the relation is not such. The sixth C function has a binary relation as input parameter and establishes whether it is a mathematical function; if it is not, then the element violating the property will be printed to the screen, otherwise a message will be printed to the screen indicating whether the function is injective, surjective, or bijective. [The project can be submitted also by first-year students.]

Scrivere una libreria ANSI C che gestisce le relazioni binarie esportando le seguenti funzioni. La prima funzione C restituisce una relazione binaria acquisita da tastiera. La seconda funzione C ha come parametro di ingresso una relazione binaria e la stampa a video. La terza funzione C ha come parametro di ingresso una relazione binaria e stabilisce se essa è una relazione d'ordine parziale, stampando a video quale proprietà non vale nel caso la relazione non sia tale. La quarta funzione C ha come parametro di ingresso una relazione binaria e stabilisce se essa è una relazione d'ordine totale, stampando a video quale proprietà non vale nel caso la relazione non sia tale. La quinta funzione C ha come parametro di ingresso una relazione binaria e stabilisce se essa è una relazione d'equivalenza, stampando a video quale proprietà non vale nel caso la relazione non sia tale. La sesta funzione C ha come parametro di ingresso una relazione binaria e stabilisce se essa è una funzione matematica; se non lo è, allora si stamperà a video quale elemento violi la proprietà, altrimenti si stamperà a video un messaggio che indica se la funzione è iniettiva, suriettiva o biiettiva. [Il progetto può essere consegnato anche da studenti del primo anno.]

2 Analisi del Problema

2.1 Input

1. Per l'acquisizione come input abbiamo una relazione binaria del tipo (a,b) che viene acquisita da tastiera;
2. Come input per le altre 5 funzioni abbiamo una relazione (precedentemente esportata dalla prima).

2.2 Output

1. Il primo problema (problema dell'acquisizione) ci richiede di andare a prendere una relazione binaria immessa tramite tastiera da un'utente;
2. Il secondo problema (problema della stampa) ci richiede di andare a stampare la relazione binaria immessa precedentemente dall'utente;
3. Il terzo problema (problema della verifica dell'ordine parziale) ci richiede di controllare se la relazione binaria immessa precedentemente dall'utente una relazione d'ordine parziale, e nel caso in cui non lo sia di andare a stampare a video le varie proprietà che non rispetta;
4. Il quarto problema (problema della verifica dell'ordine totale) ci richiede di controllare se la relazione binaria immessa precedentemente dall'utente una relazione d'ordine totale, e nel caso in cui non lo sia di andare a stampare a video le varie proprietà che non rispetta;
5. Il quinto problema (problema della verifica dell'ordine di equivalenza) ci richiede di controllare se la relazione binaria immessa precedentemente dall'utente una relazione di equivalenza, e nel caso in cui non lo sia di andare a stampare a video le varie proprietà che non rispetta;
6. Il sesto problema (problema della verifica della funzione) ci richiede di controllare se la relazione binaria immessa precedentemente dall'utente una funzione, e nel caso in cui non lo sia di andare a stampare a video le varie proprietà che non rispetta, mentre in caso in cui sia una funzione di controllare se tale funzione rispetti le proprietà di suriettività e iniettività, stampando a video se la funzione è suriettiva, iniettiva o biiettiva;

3 Progettazione dell' Algoritmo

3.1 Teoria

Per lo sviluppo di questo programma si necessita di alcuni cenni di Teoria degli insiemi quali:

Concetto di Relazione Binaria : una relazione binaria è un sottoinsieme del prodotto cartesiano di due insiemi (i quali potrebbero pure coincidere, ma ciò non è garantito).

Concetto di Relazione d' Ordine Parziale: In matematica, più precisamente in teoria degli ordini, una relazione d'ordine o ordine su di un insieme è una relazione binaria tra elementi appartenenti all'insieme che gode delle seguenti proprietà:

riflessiva

antisimmetrica

transitiva.

Concetto di Relazione d' Ordine Totale: Una relazione d' ordine si dice Totale, quando oltre a essere parziale soddisfa anche la proprietà di Dicotomia (tutti gli elementi devono essere in relazione con ogni altro elemento presente).

Concetto di riflessività : In logica e in matematica, una relazione binaria R in un insieme X è detta riflessiva se ogni elemento di X è in tale relazione con se stesso.

Concetto di transitività: In matematica, una relazione binaria R in un insieme X è transitiva se e solo se per ogni a, b, c appartenenti ad X , se a è in relazione con b e b è in relazione con c , allora a è in relazione con c .

Concetto di simmetricità: In matematica, una relazione binaria R in un insieme X è simmetrica se e solo se, presi due elementi qualsiasi a e b , vale che se a è in relazione con b allora anche b è in relazione con a .

Un sottoinsieme f di $A \times B$ è una funzione se ad ogni elemento di A viene associato da f al più un elemento di B , dando luogo alla distinzione tra funzioni totali e parziali (a seconda che tutti o solo alcuni degli elementi di A abbiano un corrispondente in B) e lasciando non specificato se tutti gli elementi di B siano i corrispondenti di qualche elemento di A oppure no.

Concetto di Iniettività: ad ogni elemento del codominio corrisponde al più un elemento del dominio, cioè elementi diversi del dominio vengono trasformati in elementi diversi del codominio.

Concetto di Suriettività: Una funzione si dice suriettiva quando ogni elemento del codominio viene raggiunto da un elemento del dominio.

3.2 Funzioni per l'acquisizione:

acquisizione() : per acquisire la relazione.

3.3 Funzioni per la verifica delle proprietà:

check_iniettivita() : per controllare se l' iniettività è rispettata o meno (0 non c' è, 1 c' è).

check_transitivita() : per controllare se la transitività viene rispettata o meno (0 non c' è, 1 c' è).

check_antisimmetria() : per controllare se l' antisimmetria viene rispettata o meno (0 non c' è, 1 c' è).

check_simmetria() : per controllare se la simmetria viene rispettata o meno (0 non c' è, 1 c' è).

check_riflessivita() : per controllare se la riflessività viene rispettata o meno (0 non c' è, 1 c' è).

check_dicotomia() : per verificare se la dicotomia viene rispettata o meno (0 non c' è, 1 c' è).

check_suriettivita(): verifica se la funzione gode della proprietà di suriettività, in questo caso sarà sempre settata a 1 in quanto tutti gli elementi del codominio (presi come gli elementi dei vari secondi termini digitati durante l' acquisizione) avranno sempre un elemento del dominio associato(dato che non si può acquisire il secondo termine se non se ne acquisisce prima il relativo primo, o arrivare alla funzione check_suriettivita() avendo acquisito solo il primo).

3.4 Funzioni principali:

`ordine_parziale()` : richiama le funzioni delle proprietà e controlla se c' è un ordine parziale(stampa a video se c' è o meno un ordine parziale, e nel caso non c' è stampa quali proprietà non vengono rispettate).

`ordine_totale()`: richiama la funzione `ordine_parziale` e `check_dicotomia` e controlla se c' è un ordine totale(stampa a video se esiste o meno un ordine totale, e nel caso non c' è stampa quali proprietà non vengono rispettate).

`relazione_equivalenza()` : richiama le funzioni delle proprietà e controlla se c' è una relazione d'equivalenza(stampa a video se c' è o meno una relazione d'equivalenza, e nel caso non c' è stampa quali proprietà non vengono rispettate).

`check_funzione()`:verifica se la relazione è una funzione(stampa a video se c' è o non c' è una funzione e nel caso non ci sia dice quale coppia non soddisfa le proprietà).

3.5 Input

Per l' input abbiamo necessità di usare una struttura dati dinamica, nella quale andiamo a salvare la Relazione Binaria dataci dall' utente, il numero delle coppie e il tipo di input (numerico o per stringhe).

L input dovrà essere dotato di diversi controlli, se l' utente sceglie di inserire un input di tipo numerico allora non potrà digitare stringhe e/o caratteri speciali etc.

La scelta di due tipi di input differente dovrà essere data per dare la possibilità all' utente nel caso scelga di fare un' input di tipo numerico di poter effettuare operazioni non legate alle funzioni della libreria, (esempio : l' utente vuole decidere di moltiplicare l' input per due, e vedere se mantiene le proprietà, con un' input di tipo numerico l' utente pu farlo e ci avrebbe un senso, con un' input di tipo stringa meno).

La scelta dell' input di tipo stringa dovrà essere data per aver maggior completezza, una relazione binaria non deve essere forzosamente numerica ma pu essere anche tra cose, oggetti, animali, colori e qualsiasi altra cosa possa venire in mente.

Alle varie funzioni verrà data come input la struttura dati salvata in precedenza dalla funzione Acquisizione, per poterne verificare le varie proprietà.

3.6 Output - Acquisizione

Durante l' acquisizione avremo diversi output video che guideranno l' utente nell' inserimento dei dati, e che segnaleranno eventuali errori commessi. Finita l' acquisizione dovremo restituire l' indirizzo della struttura, che all' interno quindi conterra' i dati inseriti dall' utente. Abbiamo scelto di fare ci perchè non essendo permesso l' utilizzo di variabili globali, il modo pi semplice di passare i dati inseriti da una funzione all' altra è quello di creare una struttura dinamica. Una volta restituito l' indirizzo della struttura, a seconda della funzione lanciata nel file Test.c si lanceranno le altre 5 funzioni, dato che queste prendono tutte in pasto l' output della prima (cioè l' indirizzo della struttura della relazione binaria) e la utilizzano per verificarne varie proprieta' .

3.7 Output - stampa

La funzione stampa avra' come output la stampa a video della struttura acquisita, con qualche aggiunta grafica(le parentesi e le virgole) per rendere il tutto pi facilmente interpretabile e leggibile.

3.8 Output - ordine_parziale

La funzione ordine_parziale avra' come output la stampa a video del risultato della verifica delle proprieta' di riflessivita' antisimmetria e transitivita' . Nel caso in cui siano tutte verificate si stampera' che la relazione è una relazione di ordine parziale, mentre nel caso in cui non siano verificate si stampera' che non lo è e il perchè (cioè quale proprieta' non è o non sono verificate).

3.9 Output - ordine_totale

La funzione ordine_totale avra' come output la stampa a video del risultato della verifica delle proprieta' necessarie ad avere una relazione d' ordine parziale, e verifichera' poi se anche la dicotomia è valida per la relazione o meno. Nel caso in cui tutto sia positivo, allora si stampera' che la relazione è di ordine totale, mentre se non lo è si stampera' cosa fa in modo che non lo sia.

3.10 Output - relazione_equivalenza

La funzione `relazione_equivalenza` avra' come output la stampa a video del risultato della verifica delle proprieta' di riflessivita' simmetria e transitivita' e nel caso in cui siano tutte positive si stampera' che la relazione è una relazione di equivalenza, mentre nel caso in cui qualcosa non sia verificato si stampera' cio' che impedisce alla relazione di essere una relazione d'equivalenza.

3.11 Output - check_funzione

La funzione `check_funzione` avra' come output la stampa a video della verifica della proprieta' che rende la relazione binaria una funzione, e in caso lo sia anche se questa è suriettiva (che poi spiegheremo essere sempre verificata) e iniettiva, e in caso sia entrambe si stampera' che la relazione binaria oltre ad essere una funzione è una funzione biiettiva.

4 Implementazione dell' algoritmo

4.1 Libreria

```
1
2  /******STRUTTURA relBin
   ******/
3  /****** Creo una struttura dove salvare le coppie
   appartenenti alla Relazione******/
4
5  typedef struct relBin{
6    /****** Coppia Numerica ******/
7    double *primo_termine ,
8           *secondo_termine ;
9
10   /****** Coppia Qualsiasi******/
11   char **prima_stringa ,
12         **seconda_stringa ;
13
14   /***** Variabili per salvare se ho acquisito una
   coppia numerica o no e il numero delle coppie****
   */
15   int controllo ,
16       dimensione ,
17       insieme_a ,
18       insieme_b ;
19 }rel_bin ;
20
21 extern rel_bin acquisizione (rel_bin);
22 extern int controllo_simmetria (rel_bin);
23 extern int controllo_riflessivita (rel_bin);
24 extern int controllo_transitivita (rel_bin);
25 extern int controllo_suriettivita (rel_bin);
26 extern void controllo_biiettivita (rel_bin);
27 extern int controllo_antisimmetria (rel_bin);
28 extern void controllo_funzione (rel_bin);
29 extern void relazione_equivalenza (rel_bin);
30 extern void ordine_totale (rel_bin);
31 extern int ordine_parziale (rel_bin);
32 extern void stampa (rel_bin);
```

4.2 Test

```
1 #include<stdio.h>
2 #include"librerie/progetto.h"
3
4 int main (void){
5     struct relBin RelazioneBinaria;
6     int scelta;
7     int scan;
8     int test_terminati;
9     scan = 0;
10    test_terminati = 0;
11    printf ("\n_Programma_per_effettuare_i_Test_sulla_
        libreria\n");
12
13
14    printf ("\n\n_Digita_il_numero_corrispondente_all_
        azione_che_si_vuole_svolgere\n");
15    printf ("\n1)_Test_Acquisizione\n2)_Esci\n");
16
17
18    while ((scelta < 1) || (scelta > 2) || scan != 1){
19        printf("\n_scelta:_");
20        fflush(stdin);
21        scan = scanf("%d",&scelta);
22    }
23    if (scelta == 1)
24        RelazioneBinaria = acquisizione(RelazioneBinaria);
25
26    if (scelta == 2){
27        printf ("\n\n.....Test_terminati.....\n\n");
28        test_terminati = 1;
29    }
30
31    scelta = -1;
32    while(scelta != 7 && test_terminati != 1){
33        printf("\n\n_Digita_il_numero_corrispondente_all_
        azione_che_si_vuole_svolgere\n");
34        printf("\n1)_Test_Acquisizione\n2)_Test_Stampa\n
        3)_Test_verifica_ordine_parziale\n4)_Test_
        verifica_ordine_totale");
35        printf("\n5)_Test_verifica_relazione_d'equivalenza\
        n_6)_Test_funzione\n7)_Esci\n");
36        scelta = -1;
```

```

37  while((scelta < 1) || (scelta > 7) || scan != 1){
38      printf("\n_scelta:_");
39      fflush(stdin);
40      scan = scanf("%d",&scelta);
41  }
42
43
44  if (scelta == 1)
45      RelazioneBinaria = acquisizione (RelazioneBinaria)
46      ;
47  if (scelta == 2)
48      stampa (RelazioneBinaria);
49  if (scelta == 3)
50      ordine_parziale (RelazioneBinaria);
51  if (scelta == 4)
52      ordine_totale (RelazioneBinaria);
53  if (scelta == 5)
54      relazione_equivalenza (RelazioneBinaria);
55  if (scelta == 6)
56      controllo_funzione (RelazioneBinaria);
57  if (scelta == 7){
58      printf (" \n\n..... Test_terminati..... \n\n");
59      test_terminati = 1;
60  }
61  return(0);
62
63  }

```

4.3 Makefile

```
Test.exe: Test.c Makefile
    gcc -ansi -Wall -O Test.c -o Test.exe
pulisci:
    rm -f Test.o
pulisci_tutto:
    rm -f Test.exe Test.o
```

5 Testing del programma

5.1 Test 1:

Test di Relazione d'ordine Totale.

Inputs: (a,a)(a,b)(b,b)

Outputs: checkriflessività : 1, checksimmetria : 0, checktransitività : 1
checkdicotomia : 1, la relazione è una relazione d'ordine totale in quanto è
rispetta anche la proprietà di Dicotomia.

```
6> Test funzione
7> Esci

scelta: 2

La relazione binaria e':
<(a,a);(a,b);(b,b) >

... Stampa Terminata ...

Digita il numero corrispondente all'azione che si vuole svolgere
1) Test Acquisizione
2) Test Stampa
3) Test verifica ordine parziale
4) Test verifica ordine totale
5) Test verifica relazione d'equivalenza
6) Test funzione
7) Esci

scelta: _
```

```
La relazione:
e' riflessiva
e' asimmetrica
e' transitiva

Quindi e' una relazione d'ordine parziale

... Controllo Ordine Parziale Terminato ...

e' dicotomica

Quindi e' una relazione d'ordine totale
... Controllo Ordine Totale Terminato ...
```


5.2 Test 2:

Test di Relazione d'ordine Parziale.

Inputs:(a,a)(b,b)(a,b)(c,c)

Outputs:checkriflessività : 1, checksimmetria : 0, checktransitività : 1 la relazione è una relazione d'ordine parziale in quanto rispetta le proprietà.

```
6> Test funzione
7> Esci

scelta: 2

La relazione binaria e':
<(a,a);(a,b);(b,b);(c,c) >

... Stampa Terminata ...

Digita il numero corrispondente all'azione che si vuole svolgere
1> Test Acquisizione
2> Test Stampa
3> Test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> Test funzione
7> Esci

scelta:
```

```
2> Test Stampa
3> Test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> Test funzione
7> Esci

scelta: 3

La relazione:
e' riflessiva
e' asimmetrica
e' transitiva

Quindi e' una relazione d'ordine parziale

... Controllo Ordine Parziale Terminato ...
```

5.3 Test 3:

Test di Relazione d'ordine non Parziale.

Inputs:(a,a)(b,b)(c,c)(d,d)(e,e)(a,b)(b,c)

Outputs:checkriflessività : 1, checksimmetria : 0, checktransitività : 0 la relazione non è una relazione d'ordine parziale in quanto non rispetta le proprietà.

```
6> Test funzione
7> Esci

scelta: 2

La relazione binaria e':

<(a,a);<(b,b);<(c,c);<(d,d);<(e,e);<(a,b);<(b,c) >

... Stampa Terminata ...

Digita il numero corrispondente all'azione che si vuole svolgere

1) Test Acquisizione
2) Test Stampa
3) Test verifica ordine parziale
4) Test verifica ordine totale
5) Test verifica relazione d'equivalenza
6) Test funzione
7) Esci

scelta:
```

```
6> Test funzione
7> Esci

scelta: 3

La relazione:

e' riflessiva
e' asimmetrica
non e' transitiva

Non e' una relazione d'ordine parziale in quanto non rispetta tutte le proprietà
,
manca la proprietà di transitività

... Controllo Ordine Parziale Terminato ...

Digita il numero corrispondente all'azione che si vuole svolgere
```

5.4 Test 4:

Test di Relazione d'equivalenza.

Inputs:(a,a)(a,b)(b,a)(b,b)

Outputs:checkriflessività : 1, checksimmetria : 1, checktransitività : 1 checkdicotomia : 0, la relazione è una relazione d'equivalenza in quanto rispetta le proprietà.

```
6> test funzione
7> Esci

scelta: 2

La relazione binaria e':
<(a,a);(a,b);(b,a);(b,b)>

... Stampa Terminata ...

Digita il numero corrispondente all'azione che si vuole svolgere
1> Test Acquisizione
2> Test Stampa
3> test verifica ordine parziale
4> test verifica ordine totale
5> test verifica relazione d'equivalenza
6> test funzione
7> Esci

scelta:
```

```
3> Test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> test funzione
7> Esci

scelta: 5
e' riflessiva
e' simmetrica
e' transitiva

Quindi e' una relazione di equivalenza

Digita il numero corrispondente all'azione che si vuole svolgere
1> Test Acquisizione
2> test Stampa
3> test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> test funzione
7> Esci

scelta:
```

5.5 Test 5:

Test di Relazione non d'equivalenza.

Inputs:(a,a)(a,b)(b,c)

Outputs:checkriflessività : 0, checksimmetria : 0, checktransitività : 0 la relazione non è una relazione d'ordine d'equivalenza in quanto non rispetta le proprietà.

```
6> Test funzione
7> Esci

scelta: 2

La relazione binaria e':
<(a,a);(a,b);(b,c)>

... Stampa Terminata ...

Digita il numero corrispondente all'azione che si vuole svolgere
1> Test Acquisizione
2> Test Stampa
3> Test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> Test funzione
7> Esci

scelta:
```

```
7> Esci

scelta: 5
non e' riflessiva
e' asimmetrica
non e' transitiva

Quindi non e' una relazione di equivalenza perche' non riflessiva
Quindi non e' una relazione di equivalenza perche' non simmetrica
Quindi non e' una relazione di equivalenza perche' non transitiva

Digita il numero corrispondente all'azione che si vuole svolgere
1> Test Acquisizione
2> Test Stampa
3> Test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> Test funzione
7> Esci

scelta:
```

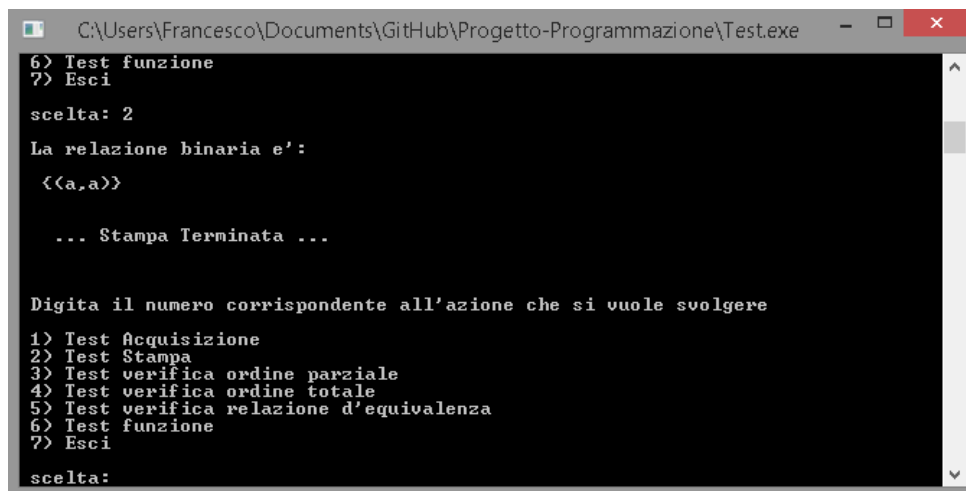
5.6 Test 6:

Test di Funzione.

Inputs:(a,a) Outputs:La relazione binaria è una funzione.

La relazione binaria è iniettiva.

La relazione binaria è biiettiva.



```
C:\Users\Francesco\Documents\GitHub\Progetto-Programmazione\Test.exe
6> Test funzione
7> Esci

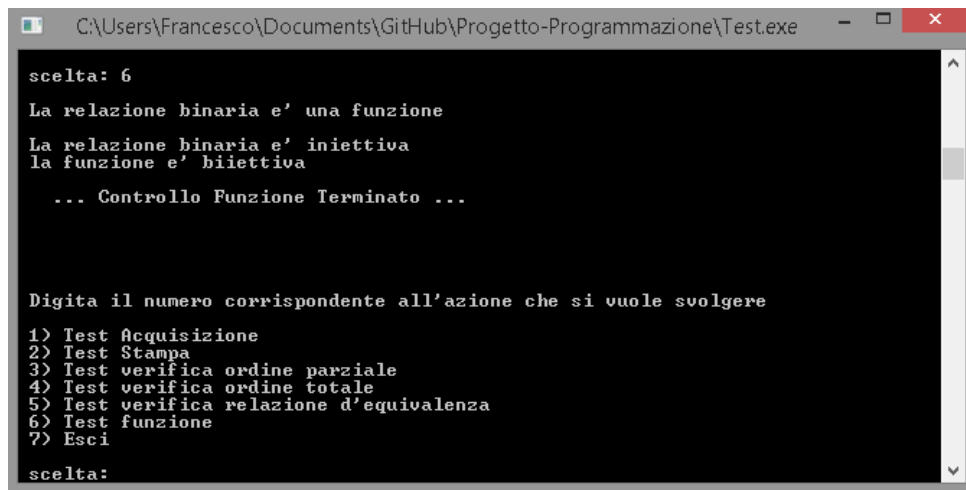
scelta: 2

La relazione binaria e':
  <<a,a>>

... Stampa Terminata ...

Digita il numero corrispondente all'azione che si vuole svolgere
1> Test Acquisizione
2> Test Stampa
3> Test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> Test funzione
7> Esci

scelta:
```



```
C:\Users\Francesco\Documents\GitHub\Progetto-Programmazione\Test.exe

scelta: 6

La relazione binaria e' una funzione
La relazione binaria e' iniettiva
la funzione e' biiettiva

... Controllo Funzione Terminato ...

Digita il numero corrispondente all'azione che si vuole svolgere
1> Test Acquisizione
2> Test Stampa
3> Test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> Test funzione
7> Esci

scelta:
```

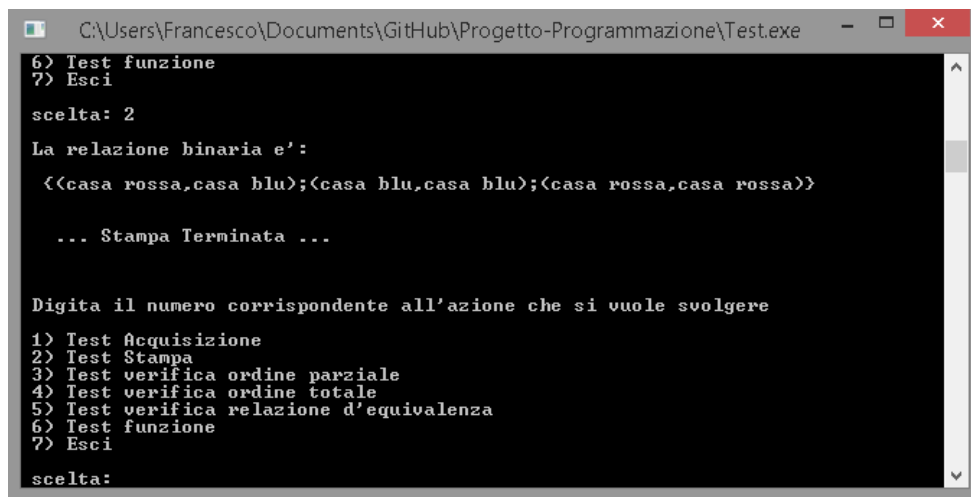
5.7 Test 7:

Test per verificare il controllo degli inputs.

Inputs:(casa rossa,casa blu)(casa blu,casa blu)(casa rossa,casa rossa)

Outputs:check_riflessività : 1,check_simmetria : 1, check_transitività : 1
dicotomia :1 la relazione è una relazione d'ordine totale in quanto rispetta le proprietà.

le funzioni funzionano anche con input contenuti degli spazi.



```
C:\Users\Francesco\Documents\GitHub\Progetto-Programmazione\Test.exe
6> Test funzione
7> Esci

scelta: 2

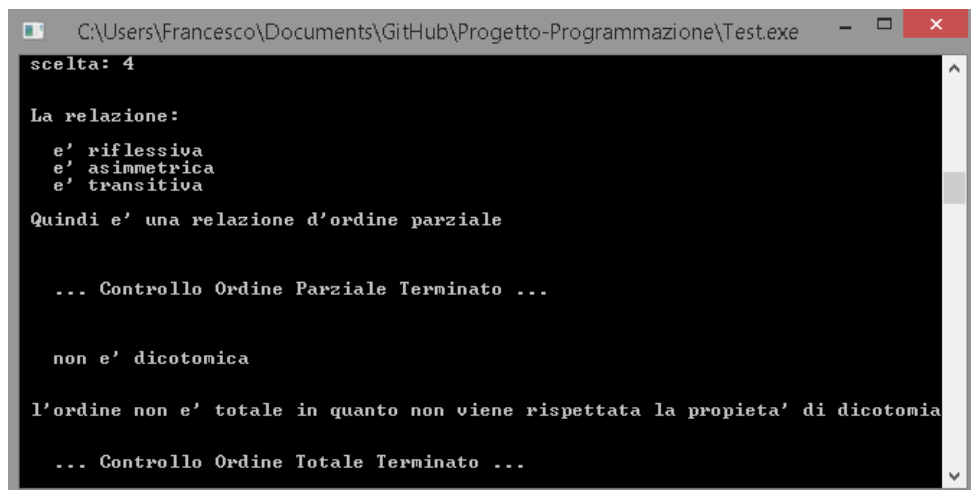
La relazione binaria e':

<(casa rossa,casa blu);(casa blu,casa blu);(casa rossa,casa rossa)>

... Stampa Terminata ...

Digita il numero corrispondente all'azione che si vuole svolgere
1> Test Acquisizione
2> Test Stampa
3> Test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> Test funzione
7> Esci

scelta:
```



```
C:\Users\Francesco\Documents\GitHub\Progetto-Programmazione\Test.exe

scelta: 4

La relazione:
e' riflessiva
e' asimmetrica
e' transitiva

Quindi e' una relazione d'ordine parziale

... Controllo Ordine Parziale Terminato ...

non e' dicotomica

l'ordine non e' totale in quanto non viene rispettata la proprieta' di dicotomia

... Controllo Ordine Totale Terminato ...
```

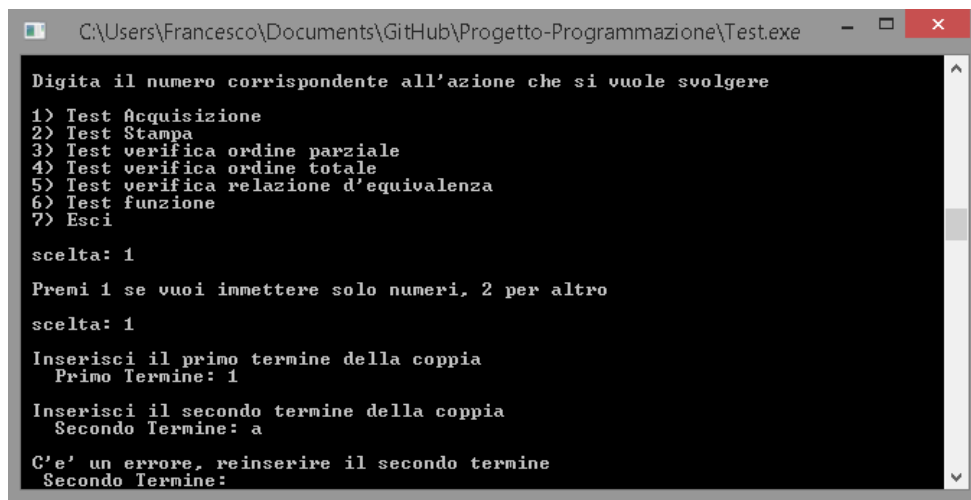
5.8 Test 8:

Test per inserire stringhe in una relazione numerica.

Inputs:(1,a)

Outputs: c' è un errore reinserisci il valore.

stampa errore in quanto si era selezionato di voler immettere un input di tipo numerico.



```
C:\Users\Francesco\Documents\GitHub\Progetto-Programmazione\Test.exe

Digita il numero corrispondente all'azione che si vuole svolgere
1> Test Acquisizione
2> Test Stampa
3> Test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> Test funzione
7> Esci

scelta: 1

Premi 1 se vuoi immettere solo numeri, 2 per altro
scelta: 1

Inserisci il primo termine della coppia
Primo Termine: 1

Inserisci il secondo termine della coppia
Secondo Termine: a

C'e' un errore, reinserire il secondo termine
Secondo Termine:
```

5.9 Test 9:

Test per vedere se una relazione binaria qualunque e' una funzione.

Inputs:(1,2)(1,1)

Outputs: La relazione binaria non è una funzione

Nel 2 elemento c'è un errore che impedisce alla relazione binaria di essere una funzione;

```
C:\Users\Francesco\Documents\GitHub\Progetto-Programmazione\Test.exe
6> Test funzione
7> Esci

scelta: 2

La relazione binaria e':
< (1.00,1.00) ; (1.00,2.00)>

... Stampa Terminata ...

Digita il numero corrispondente all'azione che si vuole svolgere
1> Test Acquisizione
2> Test Stampa
3> Test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> Test funzione
7> Esci

scelta:
```

```
C:\Users\Francesco\Documents\GitHub\Progetto-Programmazione\Test.exe
scelta: 6

Nel 2 elemento c'e' un errore che impedisce alla relazione binaria
di essere una funzione

La relazione binaria non e' una funzione

... Controllo Funzione Terminato ...

Digita il numero corrispondente all'azione che si vuole svolgere
1> Test Acquisizione
2> Test Stampa
3> Test verifica ordine parziale
4> Test verifica ordine totale
5> Test verifica relazione d'equivalenza
6> Test funzione
7> Esci

scelta:
```


5.10 Test 10:

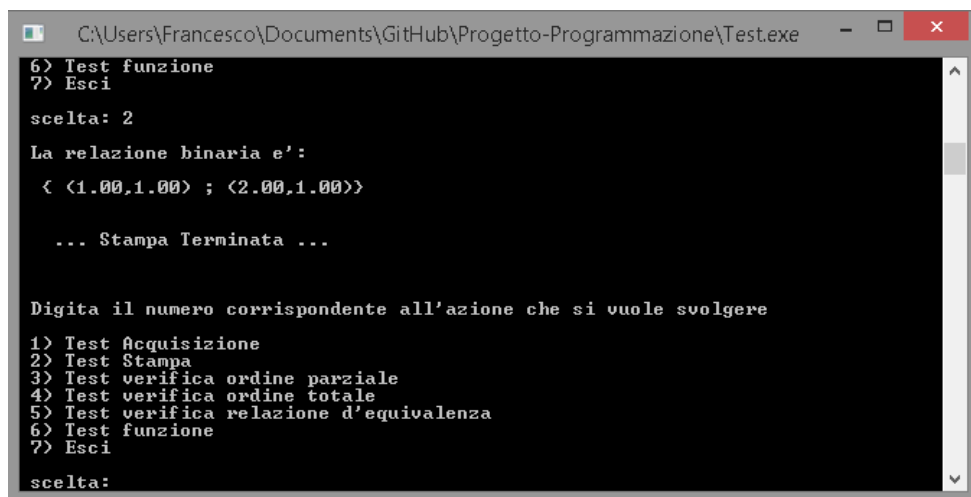
Inputs:(1,1)(2,1)

Outputs: La relazione binaria è una funzione

Nel 2 elemento c'è un errore che impedisce alla funzione di essere iniettiva

La funzione non è iniettiva

La funzione non è biiettiva



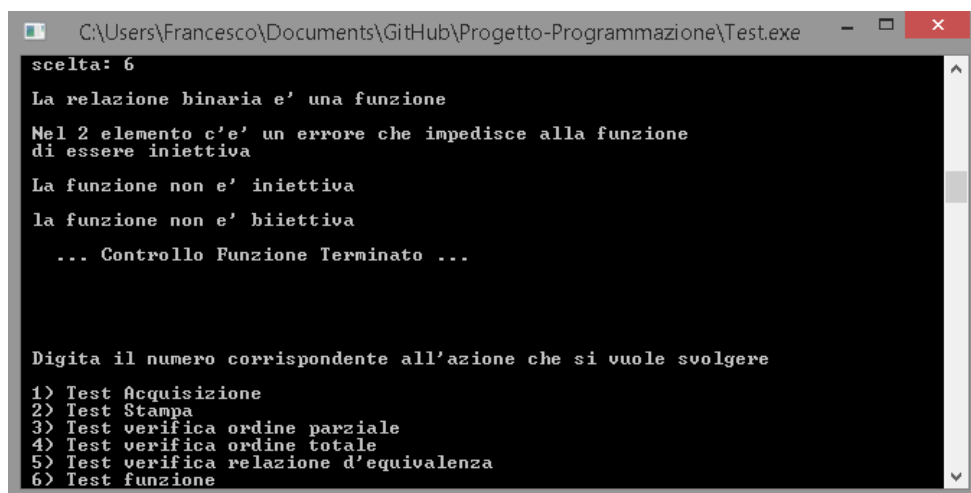
```
C:\Users\Francesco\Documents\GitHub\Progetto-Programmazione\Test.exe
6> Test funzione
7> Esci

scelta: 2

La relazione binaria e':
{ <1.00,1.00> ; <2.00,1.00> }

... Stampa Terminata ...

Digita il numero corrispondente all'azione che si vuole svolgere
1) Test Acquisizione
2) Test Stampa
3) Test verifica ordine parziale
4) Test verifica ordine totale
5) Test verifica relazione d'equivalenza
6) Test funzione
7) Esci
scelta:
```



```
C:\Users\Francesco\Documents\GitHub\Progetto-Programmazione\Test.exe
scelta: 6

La relazione binaria e' una funzione
Nel 2 elemento c'e' un errore che impedisce alla funzione
di essere iniettiva
La funzione non e' iniettiva
la funzione non e' biiettiva
... Controllo Funzione Terminato ...

Digita il numero corrispondente all'azione che si vuole svolgere
1) Test Acquisizione
2) Test Stampa
3) Test verifica ordine parziale
4) Test verifica ordine totale
5) Test verifica relazione d'equivalenza
6) Test funzione
```

6 Verica del programma

Questa porzione di codice fa in modo che una volta eseguito si abbia nel valore c la sommatoria del numero di elementi distinti inseriti dall'utente.

```
riscontro = numero_elementi
while(numero_elementi>0)
{ numero_elementi - -;
riscontro = riscontro + numero_elementi;
}
```

La postcondizione è

$$R = (\text{riscontro} = \sum_{j=0}^{\text{numero_elementi}-1} \text{numero_elementi} - j)$$

si pu rendere la tripla vera mettendo preconditione vero in quanto:

-Il predicato

$$P = (\text{numero_elementi} > 0 \wedge \text{riscontro} = \sum_{j=0}^{\text{numero_elementi}-1} \text{numero_elementi} - j)$$

e la funzione :

$$\text{tr}(\text{numero_elementi}) = \text{numero_elementi} - 1)$$

soddisfano le ipotesi del teorema dell'invariante di ciclo in quanto:

$$\{P \wedge \text{numero_elementi} > 0\} \text{riscontro} = \text{riscontro} + \text{numero_elementi}; \text{numero_elementi} = \text{numero_elementi} - -; \{P\}$$

segue da :

$$P_{\text{numero_elementi}, \text{numero_elementi}-1} \wedge \text{riscontro} \sum_{j=0}^{\text{numero_elementi}-2} \text{numero_elementi}-j$$

e denotato con P' quest'ultimo predicato, da:

$$\begin{aligned} P'_{\text{riscontro}, \text{riscontro}+\text{numero_elementi}} &= (\text{numero_elementi} > 0 \wedge \text{riscontro} + \text{numero_elementi} \\ &= \\ &= \sum_{j=0}^{\text{numero_elementi}-2} \text{numero_elementi} - j) \end{aligned}$$

$$\begin{aligned} P'_{\text{riscontro}, \text{riscontro}+\text{numero_elementi}} &= (\text{numero_elementi} > 0 \wedge c = \\ &= \sum_{j=0}^{\text{numero_elementi}-1} \text{numero_elementi} - j) \end{aligned}$$

$$\begin{aligned} \text{in quanto denotato con } P'' \text{ quest'ultimo predicato, si ha: } (P \wedge \text{numero_elementi} > 1) &= \\ (\text{numero_elementi} > 0 \wedge \text{riscontro} = \sum_{j=0}^{\text{numero_elementi}-1} \text{numero_elementi} - j \wedge \\ \text{numero_elementi} > 1) \\ | &= P'' \end{aligned}$$

* Il progresso è garantito dal fatto che $\text{tr}(\text{numero_elemnti})$ decresce di un unità ad ogni iterazione in quanto numero_elementi viene decrementata di un' unità ad ogni iterazione.

* La limitatezza segue da:

$$\begin{aligned} (P \wedge \text{tr}(\text{numero_elementi}) < 1) &= (\text{numero_elementi} > 0 \wedge c = \sum_{j=0}^{\text{numero_elementi}-1} \text{numero_elementi}-j \wedge \text{numero_elementi} > 1) \\ &\equiv (\text{riscontro} = \sum_{j=0}^{\text{numero_elementi}-1} \text{numero_elementi} - j) \end{aligned}$$

$| = \text{numero_elementi} > \text{numero_elementi} - 1$
 Poichè:

$$\begin{aligned}
 (P \wedge \text{numero_elementi} < 1) &= (\text{numero_elementi} > 0 \wedge \text{riscontro} = (P \wedge \text{numero_elementi} > 1) = \\
 (\text{numero_elementi} > 0 \wedge \text{riscontro} &= \\
 &= \sum_{j=0}^{\text{numero_elementi}-1} \text{numero_elementi} - j \wedge \text{numero_elementi} < 1) \\
 &\equiv (\text{numero_elementi} = 1 \wedge \text{riscontro} = \\
 &= \sum_{j=0}^{\text{numero_elementi}-1} \text{numero_elementi} - j \wedge \text{numero_elementi} < 1)))
 \end{aligned}$$

Dal corollario del teorema dell'invariabilità di ciclo si ha che P pu essere usato solo come preconditione dell'intera istruzione di ripetizione.

-Proseguendo infine a ritroso si ottiene prima:

$$P_{\text{numero_elementi},0} = (0 < = 0 < = \text{numero_elementi} \wedge \text{riscontro} = \sum_{j=0}^{0-1} \text{numero_elementi} - j) \text{ (riscontro} = 0)$$

e poi, denotato con P''' quest'ultimo predicato si ha:

$$P'''_{\text{riscontro},0} = (0 = 0) = \text{vero}$$