Laboratorio di Fondamenti di Informatica Lab06_2015-11-03

Anno accademico 2015/2016

1. e 2. Le cifre dei numeri

Scrivere un programma che legge da stdin un numero intero positivo (non una sequenza di caratteri, un numero! – si usi `scanf("%d", …)`) e verifica se si tratta di un numero *vario* o *ripetitivo*. Un numero è definito come **ripetitivo** se nella sua codifica decimale qualche cifra compare più di una volta, **vario** se le sue cifre sono tutte diverse. Ad esempio:

I numeri 100 121 2002 e 124565 sono **ripetitivi**

I numeri 5 124 321 e 1789 sono **vari**

Scrivere poi un programma che dati due numeri interi positivi stabilisce se sono uno l' "anagramma" dell'altro, cioè sono ottenibili uno dall'altro con una permutazione delle cifre. Esempi:

```
1234  2143  -->  anagrammi
1131  3113  -->  no
1314  1143  -->  anagrammi
 112  1122  -->  no
```

Per entrambi i problemi, come esercizio di "elasticità espressiva" si progettino (almeno) **due soluzioni**: se la prima soluzione concepita non utilizza gli array, ci si domandi se gli array possono "semplificare" la formulazione. Se la prima soluzione concepita fa uso di array, se ne sviluppi anche una che non li utilizza.

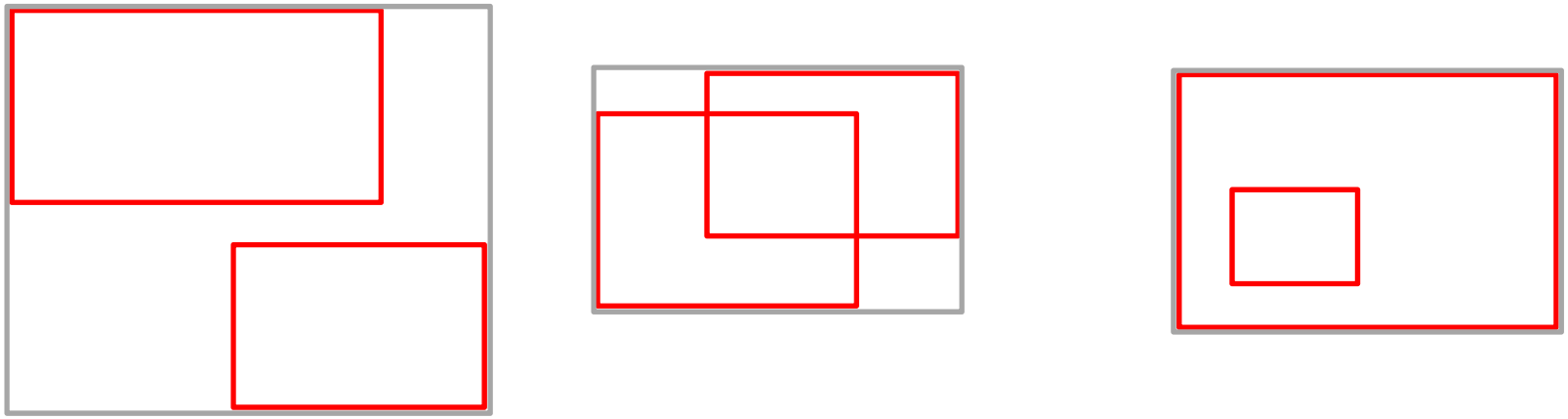
3. Triangoli e rettangoli

- Si definiscano due tipi di dato adatti a rappresentare triangoli generici e rettangoli coi lati paralleli agli assi:
  - Un triangolo sia definito come un array di tre punti nel piano
  - Un rettangolo coi lati paralleli agli assi sia **invece** definito da una **coppia** di punti (il suo vertice in alto a sinistra (NO) e il suo vertice in basso a destra (SE).
- Si scriva un programma che stabilisca se un triangolo è:
  - "ragionevolmente" isoscele (o pseudoisoscele)
  - "ragionevolmente" equilatero (o pseudoequilatero)

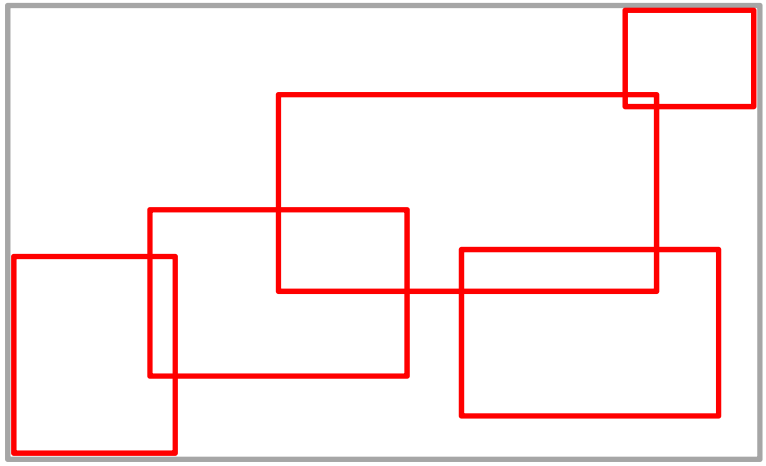
  Il programma deve verificare se valgono le relative proprietà (lunghezza dei lati) a meno di una ragionevole approssimazione in percentuale, da impostarsi come costante globale.
- Si scriva poi un programma che, dati due rettangoli coi lati paralleli agli assi:
  - Verifichi che i due rettangoli dati siano “corretti” (ovvero che il vertice di NO sia effettivamente a nord-ovest rispetto al vertice di SE o al più coincidano).
  - Calcoli il loro rettangolo di inviluppo (definito intuitivamente nella slide successiva).
  - Calcoli, se esiste, il rettangolo di intersezione, oppure segnali che i rettangoli sono interamente non sovrapposti (cioè del tutto disgiunti).

La nozione di inviluppo di rettangoli

Dati 2 generici rettangoli, il loro rettangolo di inviluppo è il minimo rettangolo che li contiene entrambi. Esempi (rettangoli in rosso, inviluppo in grigio):



La nozione, peraltro, si estende banalmente a una sequenza di N rettangoli:



4. Ribaltamento… punto per punto

Si legga da stdin una sequenza *limitata* di (*al massimo N*) caratteri, costituita da parole separate dal carattere '.' e terminata da '\n'. Si restituisca in output dapprima la sequenza ottenuta **invertendo ogni parola** ma lasciando le **parole nell'ordine originario**, e poi la sequenza ottenuta **invertendo l'ordine delle parole**, lasciando però le **parole immutate**. Si noti che è agevole acquisire la sequenza memorizzandola in una stringa con una sola istruzione `scanf("%s", …)` . Esempi:

```
Quanto.mi.piace.questo.Corso!!
otnauQ.im.ecaip.otseuq.!!osroC
Corso!!.questo.piace.mi.Quanto

Fall.leaves.after.leaves.fall
llaF.sevael.retfa.sevael.llaf
fall.leaves.after.leaves.Fall

Anna.ama.Bob.e.Ada.aveva.otto.radar
annA.ama.boB.e.adA.aveva.otto.radar
radar.otto.aveva.Ada.e.Bob.ama.Anna
```

5. Parole in ordine

Si consideri un vettore che rappresenta un elenco di N stringhe definito come segue (e come nel file in piattaforma).

```
typedef char parola[20];
typedef parola elenco[N];
```

Si stampino le parole in ordine alfabetico. Si può procedere

- provando a riordinare il vettore in modo che contenga le parole in ordine alfabetico (ma come si riordina un vettore?), oppure
- lasciando inalterato il vettore e cercando di stamparle nell'ordine giusto

Memento: la funzione `int strcmp(char s1[], char s2[])` confronta alfabeticamente s1 e s2.