mercoledì 1 febbraio 2012

Tutto il materiale in:

www.math.unipd.it/~gilberto/programmazione

Consegnare

Si consegna **sempre un solo file** che si deve chiamare "esercizio.cpp"
Messo in una cartella che contiene solo quel file (e al più a.out)
Il comando di consegna è:
consegna esame

Chi non consegna niente **viene tolto dalla lista degli iscritti** al corso
Potrà comunque riiscriversi se pensa di venire alle prossime esercitazioni

Ieri 1/2/2012 hanno consegnato 180, di questi compilano 130, quindi 50 non compilano

Gli 80 che non hanno consegnato verranno tolti dalla lista degli iscritti al corso

2 tutors (studenti magistrali di informatica) sono disposti a offrire consulenza per il corso: Scrivere a

matteo.ciman@studenti.unipd.it
stefano.bonetta@studenti.unipd.it

per accordi su come organizzare queste consulenze.

Informazioni anche a:
informatica.math.unipd.it/laurea/tutor.html

Teoria) Data la dichiarazione int A[5][10][20];

1) Specificare il tipo di A e le dimensioni dell'oggetto puntato da A;

Sia L il valore di A. Si osservi che L è semplicemente un intero (che è anche l'indirizzo RAM del primo byte del primo elemento di A).

2) Usando L, specificare il valore dell'espressione *A-3. Specificare anche il tipo di *A-3 e le dimensioni dell'oggetto puntato da *A-3.

A ha tipo int (*) [10][20] e punta ad un array int [10][20], le cui dimensioni sono 10*20*4 byte
A-3= L -3(20*4). Il tipo di *A-3 è uguale al tipo di *A che è: int (*)[20]. Punta ad un array int [20] che ha dimensioni 20*4

Programmazione) Scrivere un programma costituito da un main che includa la seguente dichiarazione: int C[5][10], B[5][10]. Senza occuparsi di mettere valori dentro C, il main deve riempire B in modo tale che alla fine del programma valga la seguente POST-condizione:
POST=($\forall a \in [0..4]$, $\forall b \in [0..9]$, B[a][b] = N. di elementi di C[a] che **non** compaiono nella colonna C[][b])

La PRE-condizione è vuota. Per ogni ciclo specificare (come commento) l'invariante e la post-condizione del ciclo.

Il programma dovrà percorrere M per righe e quindi contiene 2 cicli innestati i cui invarianti si derivano dalla POST con la ricetta di indicizzazione:

POST=($\forall a \in [0..4]$, $\forall b \in [0..9]$, B[a][b] = N. di elementi di C[a] che **non** compaiono nella colonna C[][b])

R1=($\forall a \in [0..i-1]$, $\forall b \in [0..9]$, B[a][b] = N. di elementi di C[a] che **non** compaiono nella colonna C[][b])

R2= ($\forall b \in [0..j-1]$, B[i][b] = N. di elementi di C[a] che **non** compaiono nella colonna C[][b])

```
for(int i=0;i<5; i++) //R1
 {
   for(int j=0;j<10; j++) //R2
  {

    qui si deve determinare B[i][j]



    }
POST2=(riempita B[i] correttamente)
}
POST=(riempita tutta B correttamente)
```

Calcolare B[i][j] significa calcolare quanti elementi di C[i] non sono in C[][j], si tratta di calcolare un intero che chiamiamo x:

POST 3=(x = N. di elementi di C[i][0..9] che non sono in C[][j])
Con la ricetta di indicizzazione:
R3=(x = N. di elementi di C[i][0..k-1] che non sono in C[][j])

```
int x=0;
for(int k=0; k<10; k++)
{
    decidere se C[i][k] è presente in C[][j]
}
POST3
```

decidere se C[i][k] è in C[][j] significa calcolare un booleano presente:

POST4=(presente ⇔ C[i][k] è in C[0..4][j])
da cui
R4=(presente ⇔ C[i][k] in C[0..z-1][j])
&&(0<=z<=5)

```
bool presente=false;
for(int z=0; z<5; z++) //R4
  if(C[i][k]==C[z][j])
     presente=true;
//POST4  = presente va bene
if(!presente)
 x++;
```

il programma fa calcoli inutili

possiamo uscire dal loop 4 non appena presente è vero

for(int z=0; z<5 && !presente; z++)

dimostrazione del caso di uscita