Laboratorio di Informatica

Corso di laurea triennale in Fisica

Carlo Mereghetti

Anche in questa esercitazione verranno approfonditi concetti quali:

- lettera e scrittura da/su file,
- utilizzo di array dinamici,
- utilizzo di `struct` per modellare dati,
- modularizzazione del codice.

Robot nel Piano

Il file `robot.in` contiene la descrizione di un numero imprecisato di *robot* intesi come entità puntiformi che si muovono sul piano $\mathbf{R}^2$ ad una certa velocità. La situazione di ogni robot è descritta su una riga del file da una quadrupla

$$(x(t_0), y(t_0), vx, vy)$$

ove $(x(t_0), y(t_0))$ sono le coordinate della posizione del robot al tempo t_0 rispetto ad un sistema di riferimento cartesiano assegnato, mentre vx e vy rappresentano le componenti lungo l'asse x e y, rispettivamente, del vettore velocità che consideriamo *costante nel tempo*. Ovviamente, al tempo $t_0 + \Delta t$ la nuova situazione del robot avrà coordinate

$$\begin{aligned} x(t_0 + \Delta t) &= x(t_0) + vx \cdot \Delta t \\ y(t_0 + \Delta t) &= y(t_0) + vy \cdot \Delta t, \end{aligned}$$

ed altrettanto ovviamente le componenti vx e vy rimarranno invariate.

Un tipo per definire variabili contenenti robot potrebbe essere il seguente:

```
struct robot {
     float x, y, vx, vy;
};
```

in cui i campi `x,y` mantengono le coordinate della posizione ad un certo istante di tempo mentre `vx,vy` le componenti della velocità.

ATTENZIONE!!! Nel codice del programma che viene chiesto di seguito, conviene mettere la definizione del tipo `struct robot` all'esterno del `main`, in modo che ogni funzione o prototipo definiti nel programma possa dichiarare variabili e parametri di tipo `robot`.

Tutti i file (sorgenti ed eseguibili) prodotti andranno salvati sotto la directory `Lab7`.

Il Programma

Per la gestione dei *file*, ricordate la libreria `<fstream>` la quale mette a disposizione variabili di tipo `ifstream` per aprire file in lettura e `ofstream` per aprire file in scrittura. Gli *array dinamici* si dichiarano con la sintassi `tipo *nome = new tipo[dimensione]` e si utilizzano come array tradizionali; ad esempio `nome[3]` denota la componente di posto 3 nell'array `nome`.

1. Caricate in un array dinamico di opportuna dimensione e tipo base i robot descritti nel file `robot.in` (ricordate quanto fatto alla scorsa esercitazione di laboratorio per determinare la dimensione dell'array). Contestualmente, stampate a video il numero di robot e la descrizione di ogni robot. Ad esempio, potrebbero essere stampati messaggi di questo tipo (Attenzione! 44 NON è il numero di robot presenti nel file `robot.in`, è usato solo a titolo d'esempio):

   ```
   Ci sono 44 robot cosi' distribuiti:

   Robot alla posizione (1.3,-2.3) con componenti velocita' vx = 4 e vy = 6.7

   Robot alla posizione (-7,-2.7) con componenti velocita' vx = 7.5 e vy = 1.6

   ...
   ```

2. Contate il numero di robot inclusi nel cerchio di raggio 1 centrato nell'origine al tempo iniziale t_0.

3. Salvate in un array di appoggio la nuova situazione dei robot all'istante $t_0 + \Delta t$ con $\Delta t = 1.0$ (a tal proposito, ricordate le leggi del movimento alla pagina precedente).

4. Contate il numero di robot inclusi nel cerchio di raggio 1 centrato nell'origine al tempo finale $t_0 + \Delta t$.

5. Determinate il *flusso netto* f_{in} di robot verso l'interno del cerchio. Per intenderci: contate il numero n_{out} di robot che sono nel cerchio al tempo t_0 e fuori dal cerchio al tempo $t_0 + \Delta t$, e il numero n_{in} di robot che sono fuori dal cerchio al tempo t_0 e dentro al cerchio al tempo $t_0 + \Delta t$. Stampate a video, con opportuni messaggi, i numeri $n_{\text{out}}, n_{\text{in}}$ e $f_{\text{in}} = n_{\text{in}} - n_{\text{out}}$.

6. Salvate nel file `robot.out` la situazione dei robot inclusi nel cerchio di raggio 1 centrato nell'origine al tempo finale $t_0 + \Delta t$.

Per modularizzare la struttura del codice prevedete la scrittura delle seguenti function:

- `bool nelCerchio( robot a )`: restituisce `true` se `a` giace entro il cerchio di raggio 1, `false` altrimenti;
- `robot nuovaSit( robot a, float d )`: restituisce la nuova situazione di `a` dopo `d` secondi da t_0;
- `void stampa( robot a )`: stampa a video la situazione di `a` nella forma:

  ```
  Robot alla posizione (1.3,-2.3) con componenti velocita' vx = 4 e vy = 6.7
  ```

Tali funzioni vanno scritte dopo il `main`, ricordando di riportare prima del main il loro *prototipo*.