Laboratorio di programmazione

17 gennaio 2008

Vita

Vita è un *automa cellulare* inventato dal matematico John Conway per studiare un'emulazione elementare dei processi vitali e diventato famoso dopo la sua descrizione su *Scientific American* nel 1971.

L'automa è composto da una matrice $N \times M$ di elementi, dette *cellule*, che possono essere in due stati: vive o morte. Attorno a ogni cellula ci sono otto cellule adiacenti (nord, sud, est, ovest, nord-ovest, nord-est, sud-ovest e sud-est): si noti che la matrice è pensata come una "ciambella" (il termine matematico preciso è *toro*): il bordo destro è adiacente a quello sinistro, così come il bordo superiore è adiacente a quello inferiore. Per esempio, la cella di coordinate $(0,0)$ ha a nord la cella $(0,M-1)$ e a ovest la cella $(N-1,0)$.

Le regole di evoluzione della Vita sono molto semplici: se una cellula è viva, sopravvive all'istante di tempo successivo se nel suo intorno ci sono due o tre cellule vive; altrimenti muore per solitudine o sovraffollamento. Se una cellula è morta e nel suo intorno ci sono esattamente tre cellule vive diventa viva all'istante successivo.

Dovete scrivere un programma che simuli Vita e visualizzi "graficamente" l'evoluzione di una matrice le cui dimensioni sono specificate sulla riga di comando.

Per ottenere la visualizzazione, potete stampare la matrice emettendo uno spazio per le cellule morte, e un altro carattere (per esempio `*`) per le cellule vive.

Il vostro programma deve prendere in input sulla riga di comando tre parametri: il numero $N \leq 200$, il numero $M \leq 200$ e una probabilità p tra 0 e 1 che verrà utilizzata per inizializzare la matrice: ogni cella sarà viva indipendentemente con probabilità p.

Suggerimenti

1. Per realizzare la simulazione, dovete utilizzare due matrici. La prima rappresenta la configurazione corrente. Da essa potete calcolare quella successiva, che memorizzerete nella seconda. A questo punto potrete ricopiare la seconda matrice sulla prima, visualizzarla e ricominciare daccapo. Se vi considerate particolarmente a vostro agio con array multidimensionali, potete utilizzare una matrice tridimensionale $2 \times N \times M$ e fare affidamento sull'aritmetica modulare per scambiare a ogni iterazione il ruolo dei due "strati" bidimensionali $N \times M$ della matrice, evitando così la copia.

2. Per ottenere l'effetto di considerare la matrice una ciambella, dovete incrementare o decrementare le coordinate utilizzando l'aritmetica modulare. Ad esempio, se la matrice si chiama `A`, le celle adiacenti ad `A[x][y]` sono `A[(x+1) % N][y]`, `A[(x+N-1) % N][y]`, `A[x][(y+1) % M]`, `A[x][(y+M-1) % M]`, `A[(x+1) % N][(y+1) % M]`, `A[(x+N-1) % N][(y+1) % M]`, `A[(x+1) % N][(y+M-1) % M]`, `A[(x+N-1) % N][(y+M-1) % M]`. **Attenzione:** non eliminate i `+N` e i `+M` apparentemente ridondanti; sono necessari perché l'implementazione del modulo sui numeri negativi è errata.

3. Per trasformare un argomento sulla riga di comando in un `float` potete usare la funzione `atof()`. Quindi, per convertire i tre parametri come richiesto sono sufficienti gli assegnamenti

```
N = atoi(argv[1]);
M = atoi(argv[2]);
```

```
p = atof(argv[3]);
```

4. Per stabilire se una cellula deve essere viva nello stato iniziale potete usare il test

   ```
   (rand()/(RAND_MAX+1.0) <= p)
   ```

   che è vero con probabilità `p`.

5. È utile, per quanto possibile, strutturare il programma: create una funzione `visualizza()` che stampa la matrice come richiesto in cima allo schermo, e provatela separatamente.

6. Per matrici piccole, la visualizzazione potrebbe essere troppo rapida: inserite in tal caso l'istruzione `usleep(100000);` all'interno del ciclo principale (ritarda l'esecuzione di un decimo di secondo).

7. Dato che per ottenere un effetto di pseudo-animazione è necessario sovrascrivere la visualizzazione precedente, potete usare l'istruzione `printf("\x1b[H");` per posizionare il cursore nell'angolo in alto a sinistra senza cancellare lo schermo, e l'istruzione `printf("\x1b[H\x1b[2J");` per cancellare l'intero schermo.

8. Una volta che il programma è ben collaudato, potete provare ad aumentare le dimensioni massime possibili a 1000×1000; premendo poi il tasto di destra del mouse quando il puntatore è sopra il terminale, potete scegliere un corpo molto piccolo e visualizzare animazioni di grandi dimensioni.

9. Il programma non deve terminare mai (come la vita). Potrete interromperlo comunque con Control-C (come nella fine del mondo).

10. Ovviamente, per utilizzare alcune delle funzioni suggerite sarà necessario includere opportuni file di intestazione `.h`, per scoprire quali, usate il comando `man` seguito dal nome della funzione che volete usare.