Libera Università degli Studi di Urbino

Informatica Applicata – Programmazione degli elaboratori (M. Bernardo)

Relazione per il progetto d’esame di Cappellini Giovanni #148158

1. **Specifica del problema**

Si rimanda a http://www.sti.uniurb.it/bernardo/teaching/prog_elab/progetto-2002-09-19.html

2. **Analisi del problema**

Il problema richiede di trattare una "scacchiera con 10 righe e 10 colonne"; nell'ottica del linguaggio C viene immediato pensare ad un *array* a due ordini (le righe e le colonne della scacchiera):

```
int scacchiera[10][10];
```

Nell'*array* le caselle occupate da un organismo saranno identificate da "1" e quelle vuote da "0". Ad ogni configurazione di scacchiera ne corrisponde un'altra, la "generazione successiva". La configurazione successiva di una casella viene determinata da regole precise che tengono conto dello stato delle caselle adiacenti. Sarebbe sbagliato applicare le regole per la determinazione della generazione successiva **sulla scacchiera principale:** in questo caso, al momento di processare le righe inferiori il programma si trova di fronte ad una scacchiera già contaminata. Sembra necessario riportare la configurazione madre un scacchiera temporanea (e quindi in un altro *array*). E' però possibile implementare tutto in un array tridimensionale. Per garantire una facile conversione del programma a scacchiere di diverse dimensione viene definita una costante, DIM.

```
int scacchiera[2][DIM][DIM];
```

La prima dimensione ci permette di scegliere su quale array scrivere o leggere. `scacchiera[][][]` va formattata forzando ogni cella a zero. Il numero delle generazioni successive da visualizzare è dato dall'utente. Poi bisogna controllare lo stato delle caselle adiacenti, contando quelle che contengono organismi. In base a ciò, si fanno le opportune modifiche alla casella: può morire un organismo o può nascere. La nascita di un organismo ha un margine di casualità: a questo riguardo il linguaggio ANSI C fornisce una funzione di numeri pseudo-casuali, int rand(). Sarà proprio questa funzione, definita in stdlib.h, ad essere utilizzata.

Per ricapitolare, seguono gli input e gli output del problema:

INPUT:

```
/*la configurazione madre*/
int scacchiera[DIM_SCACCHIERA][DIM_SCACCHIERA];

/*il numero di generazioni successive da visualizzare*/
int gen_succ;
```

OUTPUT:

```
/*viene visualizzata a schermo una rappresentazione dello
 *stesso array dopo l'applicazione delle regole*/
```

EXTRA:

```
/*per scorrere gli array*/
int i, j;
```

```
/*per contare le generazioni*/
int gen;
/*per contare le caselle adiacenti*/
int contatore;
/*per scegliere su quale array lavorare*/
int nuova, vecchia;
/*per il processo di input*/
int riga, colonna;
```

3. Progettazione dell'algoritmo

1) Richiesta della configurazione iniziale
2) Acquisizione della configurazione iniziale
3) Verifica della correttezza dell'input
4) In caso di incorrettezza, ripetere da 2) a 4)
5) Richiesta del numero di generazioni da elaborare (verificare la correttezza)
6) Elaborazione dei dati
7) Visualizzare le generazioni successive richieste

Raffinamento dell'algoritmo:

1) Richiesta della configurazione iniziale
2) Acquisizione della configurazione iniziale
3) Verifica della correttezza della configurazione
4) In caso di incorrettezza, ripetere da 2) a 4)
5) Richiesta del numero di generazioni da elaborare (verificare la correttezza)
6.1) Cambio dell'array da elaborare
6.2) Elaborazione dei dati
6.3) Se serve un'altra generazione, rifare da 6.1
7) Visualizzare le generazioni successive richieste

2° raffinamento dell'algoritmo:

1)Richiesta della configurazione iniziale
2)Acquisizione della configurazione iniziale
3)Verifica della correttezza della configurazione
4)In caso di incorrettezza, ripetere da 2) a 4)
5)Richiesta del numero di generazioni da elaborare
6.1)Copia nell'array temporaneo dei dati
6.2.1) Casella per casella, contare gli organismo adiacenti
6.2.2) Secondo le regole, modificare la configurazione
6.3) Se serve un'altra generazione, rifare da 6.1
7) Visualizzare le generazioni successive richieste

4. Implementazione dell’algoritmo

```
/*************************************************************************
 *Questo programma simula l'evoluzione degli organismi di una scacchiera. *
 *Progetto d'esame di Giovanni Cappellini #148158                          *
 *Corso di laurea in Informatica Applicata                                 *
 *Programmazione degli Elaboratori (3/10/2002)                             *
 *Professore: Marco Bernardo                                               *
 *************************************************************************/

#include <stdio.h>

#include <stdlib.h> /*questa libreria contiene la funzione rand()*/

#define DIM 12 /*questa costante rappresenta la dimensione della scacchiera*/
/*per semplificare l'algoritmo si usa una scacchiera con un contorno di caselle infertili*/

/*Questa funzione si occupa di stampare a schermo eventuali istruzioni*/
void istruzioni(void);

/*Questa funzione scambia i valori di due variabili (interi)*/
void scambia(int a, int b);

/*Questa funzione rappresenta il secondo input*/
int acquisisci_generazioni(void);

int main(void)
{
       int scacchiera[2][DIM][DIM]; /*la prima dimensione permette di scegliere su quale
scacchiera scrivere*/
    char conferma; /*Per conoscere la volonta' di sapere le istruzioni*/
       int gen_succ, gen; /*Il numero delle configurazioni da elaborare*/
       int i, j; /*Per scorrere gli array*/
       int contatore; /*Per contare le caselle vuote (o piene) adiacenti*/
       int nuova = 0, vecchia = 1; /*per scegliere su quale scacchiera scrivere*/
       int riga, colonna;

       /*inizializzazione degli array*/
       for (i = 0; i < DIM; i++)
       {
         for (j = 0; j < DIM; j++)
              scacchiera[0][i][j] = 0;
    }

    for (i = 0; i < DIM; i++)
       {
         for (j = 0; j < DIM; j++)
              scacchiera[1][i][j] = 0;
    }

    /*A questo punto si richiede all'utente se desidera visualizzare le istruzioni*/
       printf("Si desidera visualizzare le istruzioni? (premere S per si\' o N per no): ");
       scanf(" %c", &conferma);

       /*In caso di conferma, viene eseguita la funzione "istruzioni()"*/
        if ((conferma == 's') || (conferma == 'S'))
              istruzioni();

       /*input: configurazione della scacchiera*/
       printf("\nCONFIGURAZIONE DELLA SCACCHIERA (digitare 0 0 per terminare):\n");

       do
       {
           printf("Riga e colonna: ");
           scanf("%d%d", &riga, &colonna);
           if (riga == 0 && colonna == 0)
                printf("\nInput terminato.\n");
           else if (riga >= DIM -1 || riga < 0 || colonna >= DIM - 1 || colonna < 0)
             printf("\n***ERRORE NELL'INPUT*** Riprovare o digitare 0 0 per terminare.\n\n");
           else
                scacchiera[nuova][riga][colonna] = 1;
    }
    while (riga != 0 && colonna != 0);
```

```
      /*input: richiesta del numero di generazioni da elaborare*/
      gen_succ = acquisisci_generazioni();

      /*a questo punto c'e' un while per produrre tante nuove generazioni quante richieste*/
      printf("\n");
   gen = 0;
      conferma = 's';

      while ((gen != gen_succ) && ((conferma == 's') || (conferma == 'S')))
      {
          scambia(nuova, vecchia);

             /*in pratica e' una scacchiera "ritagliata" dalla prima*/
             /*ogni casella adesso ha caselle adiacenti*/
             for (i = 1; i < DIM - 1; i++)
             {
                    for (j = 1; j < DIM - 1; j++)
                    {
                       contatore = 0;
              if (scacchiera[vecchia][i - 1][j - 1])
                  contatore++;
                       if (scacchiera[vecchia][i - 1][j])
                        contatore++;
                     if (scacchiera[vecchia][i - 1][j + 1])
                           contatore++;
                       if (scacchiera[vecchia][i][j - 1])
                           contatore++;
                       if (scacchiera[vecchia][i][j + 1])
                           contatore++;
                       if (scacchiera[vecchia][i + 1][j - 1])
                        contatore++;
                       if (scacchiera[vecchia][i + 1][j])
                           contatore++;
                       if (scacchiera[vecchia][i + 1][j + 1])
                           contatore++;

              if (scacchiera[vecchia][i][j])
                       {
                                 /*morte per sovraffolamento o solitudine*/
                                 if ((contatore >= 4) || (contatore < 2))
                                        scacchiera[nuova][i][j] = 0;
                                 else
                                      scacchiera[nuova][i][j] = 1;
                       }
                       else
                       {
             /*nascita di un nuovo organismo*/
/*l'elemento casuale che determina la nascita di un nuovo organismo e' dato dalla*
 *funzione pseudocasuale rand()                                                   */
                                 if (contatore == 3 && rand() % 2 == 0)
                                        scacchiera[nuova][i][j] = 1;
                                 else
                                      scacchiera[nuova][i][j] = 0;
                       }
                   }
             }
             /*stampa a schermo della nuova generazione*/
             printf("\nGenerazione %d:\n", gen);
             for (i = 1; i < DIM - 1; i++)
             {
             /*l'if serve solo per la corretta formattazione dell'output*/
                    if (i >= DIM - 2)
              printf("Riga %d: ", i);
                    else
              printf("Riga  %d: ", i);
                    for (j = 1; j < DIM - 1; j++)
                           printf("%d ", scacchiera[nuova][i][j]);
                    printf("\n");
             }
             gen++;

             if (gen != gen_succ)
             {
```

```
                    printf("\nProcedere con la generazione %d? (S/N)", gen);
                    scanf(" %c", &conferma);
                    printf("\n");
             }
       }
       return(0);
}

       /*La funzione "Istruzioni" non ha alcun parametro ne' return, stampa solo a video*/
void istruzioni()
{
       printf("\nViene considerata una scacchiera con 10 righe e colonne.\n");
       printf("Ogni casella puo\' essere libera oppure puo\' contenere un organismo.\n");
       printf("Ogni casella, escluse quelle ai bordi, e\' considerata avere 8 caselle
adiacenti,");
       printf("che sono quelle che la circondano.\n");
       printf("Data una generazione di organismi, ovvero una configurazione della
scacchiera,\n");
       printf("la generazione successiva viene determinata in base alle seguenti regole:\n");
       printf("un organismo puo\' nascere in ogni casella vuota ");
       printf("che ha esattamente\ntre caselle adiacenti contenenti un organismo;\n");
       printf("un organismo con quattro o piu\' organismi adiacenti muore per
sovraffollamento;\n");
       printf("un organismo con meno di due organismi adiacenti muore di solitudine;\n");
       printf("un organismo con due o tre organismi adiacenti sopravvive.\n");
       printf("1 indica presenza dell\'organismo\n");
       printf("0 indica assenza dell\'organismo\n");
       /*Istruzioni per l'acquisizione della configurazione iniziale*/
       printf("La configurazione della scacchiera avviene digitando le coordinate di ogni\n");
    printf("casella che si desidera far occupare da un organismo.\n");
       printf("Le cifre devono essere spaziate. ESEMPIO: 1 1\n");
}

void scambia (int a, int b)
{
    int tmp;
    tmp = b;
    b = a;
       a = tmp;
}

int acquisisci_generazioni(void)
{
    int a;

    do
       {

        printf("\nDigitare il numero delle generazioni da elaborare: ");
           scanf("%d", &a);
           if (a < 0)
                printf("\n***ERRORE NELL'INPUT*** Le generazioni non possono essere minori di
zero!");
    }
    while (a < 0);

    return(a);
}
```

5. Testing del programma

Dopo aver compilato con il comando (contenuto all'interno del makefile):

gcc –ansi esame.c –o esame

si ottiene l'eseguibile.

Il programma chiede se si vogliono visualizzate a schermo la visualizzazione delle istruzioni, e chiede la risposta.

```
SIMULAZIONE DELLA VITA DEGLI ORGANISMI DI UNA SCACCHIERA
Si desidera visualizzare le istruzioni? (premere S per si' o N per no):
```

A questo punto battiamo "s" (che sta per sì) sulla tastiera e ci vengono così elencate le istruzio ni del programma: cosa fa e come utilizzarlo. Il prossimo input richiesto è quello delle coordinate della prima casella che vogliamo riempire con un organismo:

```
CONFIGURAZIONE DELLA SCACCHIERA (digitare 0 0 per terminare):
Riga e colonna:
```

Come primo esperimento diamo subito 0 0 per una scacchiera totalmente priva di forme di vita. Il programma si accontenta anche di questo visualizzando,

```
Input terminato.
```

e chiede quante generazioni si desidera elaborare.

```
Digitare il numero delle generazioni da elaborare:
```

Proviamo con 10. Il risultato sono 10 scacchiere come questa:

```
Generazione 0:
Riga  1: 0 0 0 0 0 0 0 0 0 0
Riga  2: 0 0 0 0 0 0 0 0 0 0
Riga  3: 0 0 0 0 0 0 0 0 0 0
Riga  4: 0 0 0 0 0 0 0 0 0 0
Riga  5: 0 0 0 0 0 0 0 0 0 0
Riga  6: 0 0 0 0 0 0 0 0 0 0
Riga  7: 0 0 0 0 0 0 0 0 0 0
Riga  8: 0 0 0 0 0 0 0 0 0 0
Riga  9: 0 0 0 0 0 0 0 0 0 0
Riga 10: 0 0 0 0 0 0 0 0 0 0
```

Il programma termina così la sua esecuzione.

2) Configurando come input una scacchiera "sparsa" (per esempio con organismi solo agli ango li):

```
Riga e colonna: 1 1
Riga e colonna: 1 10
Riga e colonna: 10 1
Riga e colonna: 10 10
```

Si ottiene lo stesso output di prima.

3) Per verificare la nascita di nuovi organismi, lasciamo vuote le caselle agli angoli e gli diamo tre organismi adiacenti:

```
Riga e colonna: 1 2
Riga e colonna: 2 1
Riga e colonna: 2 2
Riga e colonna: 9 10
Riga e colonna: 9 9
Riga e colonna: 10 9
Riga e colonna: 1 9
Riga e colonna: 2 10
Riga e colonna: 2 9
Riga e colonna: 9 1
Riga e colonna: 9 2
Riga e colonna: 10 2
```

Non è bastato avere 3 vicini: la nascita infatti deve coadiuvata dalla casualità.

4) Riproviamo riempiendo la prima riga e lasciando vuota il resto della scacchiera:

```
Riga e colonna: 1 1
Riga e colonna: 1 2
Riga e colonna: 1 3
Riga e colonna: 1 4
Riga e colonna: 1 5
Riga e colonna: 1 6
Riga e colonna: 1 7
Riga e colonna: 1 8
Riga e colonna: 1 9
Riga e colonna: 1 10
```

Ecco l'output che si ottiene: sono nati alcuni organismi sulla seconda riga (dove tutte le caselle, eccetto quelle agli estremi, avevano tre vicini:

```
Riga  1: 0 1 1 1 1 1 1 1 1 0
Riga  2: 0 0 0 1 1 0 1 1 1 0
Riga  3: 0 0 0 0 0 0 0 0 0 0
Riga  4: 0 0 0 0 0 0 0 0 0 0
Riga  5: 0 0 0 0 0 0 0 0 0 0
Riga  6: 0 0 0 0 0 0 0 0 0 0
Riga  7: 0 0 0 0 0 0 0 0 0 0
Riga  8: 0 0 0 0 0 0 0 0 0 0
Riga  9: 0 0 0 0 0 0 0 0 0 0
Riga 10: 0 0 0 0 0 0 0 0 0 0
```

5) Come ultimo test, passiamo numeri negativi come input. Ecco il responso:

```
***ERRORE NELL'INPUT*** Riprovare o digitare 0 0 per terminare.
```