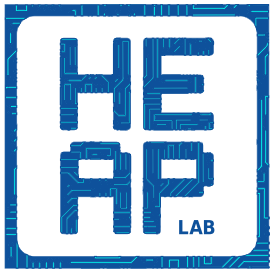
Tic-Tac-Toe in C

Stefano Cherubin*



29/11/2019

[**Informatica A**] Esercitazione #16

corso per Ing. Gestionale a.a. 2019/20

*<nome>.<cognome>@polimi.it

Indice

1 Una griglia

Scrivere una funzione C che stampi a video una griglia rettangolare di dimensioni arbitrarie.

Listato 1: stampa griglia sul terminale

```
1  #include<stdio.h>
2
3  #define M 3
4  #define N 3
5
6  void print_grid() {
7    int i,j;
8
9    // upper border
10   for(j = 0; j < N; j++)
11     printf("__");
12   printf("\n");
13
14   for(i = 0; i < M; i++) {
15     // bottom border
16     for (j = 0; j < N; j++)
17       printf("|_");
18
19     // right border and newline
20     printf("|\n");
21   }
22   return;
23 }
```

2 Pretty print di una matrice

Scrivere una funzione C che stampi a video una matrice di elementi di tipo char con opportuni contorni.

Listato 2: stampa griglia con contenuto sul terminale

```
1  #include<stdio.h>
2
3  #define M 3
4  #define N 3
5
6  char map[M][N]
7
8  void print_map() {
9    int i,j;
10
11   // upper border
12   for(j = 0; j < N; j++)
13     printf("____");
14   printf("\n");
15
16   for(i = 0; i < M; i++) {
17     // some space between border and content
18     for (j = 0; j < N; j++)
19       printf("|   ");
20
21     // right border and newline
22     printf("|\n");
23
24     // content
25     for (j = 0; j < N; j++)
26       printf("| %c ", map[i][j]);
27     // right border and newline
28     printf("|\n");
29
30     // bottom border
31     for (j = 0; j < N; j++)
32       printf("|___");
33     // right border and newline
34     printf("|\n");
35   }
36   return;
37 }
```

3 Tic-Tac-Toe

Si implementi in C il gioco tic-tac-toe.

3.1 Griglia di tic-tac-toe

Stampare a video la griglia di tic-tac-toe, con opportuno spazio per eventuali simboli lasciati dai giocatori.

Si cancelli ogni traccia di precedenti versioni della matrice dal terminale

3.1.1 Clear screen

Per pulire lo schermo esiste una sequenza di comandi che, se inviati in output al terminale, è in grado di:

- posizionare il cursore all'inizio della schermata
- pulire la schermata dalla posizione del cursore in avanti

Questi comandi sono rispettivamente \033[H e \033[J e sono supportati dai principali terminali dei sistemi Linux e Mac OS. Windows invece offre la funzione di libreria clrscr() dall'header conio.h che implementa la stessa funzionalità.

Listato 3: Tic-Tac-Toe

```
1  #include<stdio.h>
2  //#include <conio.h>
3
4  #define L 3
5
6  void print_map(char map[][L]) {
7    printf("\033[H\033[J");
8    // clrscr();
9    printf(" %c | %c | %c \n", map[0][0], map[0][1], map
       [0][2]);
10   printf("___|___|___\n");
11   printf("   |   |   \n");
12   printf(" %c | %c | %c \n", map[1][0], map[1][1], map
       [1][2]);
13   printf("___|___|___\n");
14   printf("   |   |   \n");
15   printf(" %c | %c | %c \n", map[2][0], map[2][1], map
       [2][2]);
16   return;
17 }
```

3.2 Controllare vittoria / sconfitta

Data una matrice che rappresenta lo stato del gioco, determinare se uno dei due giocatori ha vinto.

Listato 4: Tic-Tac-Toe (controllo vittoria)

```
1  #define L 3
2
3  int win(char map[][L]) {
4    int i, flag;
5    for (flag = 1, i = 0; i < L && flag; i++)
6     if (map[i][0] == map[i][1] && map[i][0] == map[i
         ][2] && map[i][0] != ' ')
7      flag = 0;
8
9    for (i = 0; i < L && flag; i++)
10    if (map[0][i] == map[1][i] && map[0][i] == map[2][i
         ] && map[0][i] != ' ')
11     flag = 0;
12
13   if (map[0][0] == map[1][1] && map[0][0] == map[2][2]
         && map[0][0] != ' ')
14     flag = 0;
15   if (map[0][2] == map[1][1] && map[0][2] == map[2][0]
         && map[0][2] != ' ')
16     flag = 0;
17
18   if(flag)
19     return 0;
20   return 1;
21 }
```

3.3 Accesso alla matrice

Semplificare l'accesso alla matrice, usando un unico valore $pos \in [1; 9]$ per indicare la posizione.

Uno switch-case funziona sempre in questi casi.

Listato 5: Tic-Tac-Toe matrix access

```
1  #define L 3
2
3  char get_elem(int pos, char map[][L]){
4    switch(pos){
5     case 1:
6      return map[0][0];
7      break;
8     case 2:
9      return map[0][1];
10     break;
11    case 3:
12     return map[0][2];
13     break;
14    case 4:
15     return map[1][0];
16     break;
17    case 5:
18     return map[1][1];
19     break;
20    case 6:
21     return map[1][2];
22     break;
23    case 7:
24     return map[2][0];
25     break;
26    case 8:
27     return map[2][1];
28     break;
29    case 9:
30     return map[2][2];
31     break;
32    default:
33     return '\0';
34     break;
35   }
36    return '\0';
37 }
```

Tuttavia è possibile ottimizzare l'accesso con un gioco di indici.

Listato 6: Tic-Tac-Toe - simplified matrix access

```
1  #define L 3
2
3  char get_elem(int pos, char map[][L]) {
4    return map[(pos-1)/L][(pos-1)%L];
5  }
6
7  void set_elem(int pos, char val, char map[][L]) {
8     map[(pos - 1) / L][(pos - 1) % L] = val;
9     return;
10 }
```

3.4 Il gioco completo - Soluzioni

Listato 7: Tic-Tac-Toe

```
1  #include<stdio.h>
2  //#include<conio.h>
3
4  #define L 3
5
6  const char X = 'x';
7  const char O = 'o';
8
9  void print_map(char map[][L]);
10 char get_elem(int pos, char map[][L]);
11 void set_elem(int pos, char val, char map[][L]);
12 int win(char map[][L]);
13
14 int main(int argc, char*argv[]){
15   char mat[L][L];
16   int i, j, k;
17   int player = 0;
18   int turn = 0;
19   for (i = 0; i < L; i++)
20     for (j = 0; j < L; j++)
21       mat[i][j] = ' ';
22
23   print_map(mat);
24   do {
25     do {
26       printf("Turno %d - dove vuoi posizionare? (1-9):
             ", turn);
27       scanf("%d", &k);
28     }while(get_elem(k, mat) != ' ');
29    set_elem(k, (player++ % 2) ? X : O, mat);
30    print_map(mat);
31    turn++;
32   } while(turn < 9 && !win(mat));
33
34   print_map(mat);
35   return 0;
36 }
37
38 int win(char map[][L]) {
39   int i, flag;
40   for (flag = 1, i = 0; i < L && flag; i++)
```

```
41      if (map[i][0] == map[i][1] && map[i][0] == map[i
           ][2] && map[i][0] != ' ')
42       flag = 0;
43
44    for (i = 0; i < L && flag; i++)
45     if (map[0][i] == map[1][i] && map[0][i] == map[2][i
           ] && map[0][i] != ' ')
46      flag = 0;
47
48    if (map[0][0] == map[1][1] && map[0][0] == map[2][2]
          && map[0][0] != ' ')
49      flag = 0;
50    if (map[0][2] == map[1][1] && map[0][2] == map[2][0]
          && map[0][2] != ' ')
51      flag = 0;
52
53    if(flag)
54     return 0;
55    return 1;
56  }
57
58  char get_elem(int pos, char map[][L]) {
59    return map[(pos-1)/L][(pos-1)%L];
60  }
61
62  void set_elem(int pos, char val, char map[][L]) {
63     map[(pos - 1) / L][(pos - 1) % L] = val;
64     return;
65  }
66
67  void print_map(char map[][L]) {
68    // clrscr();
69    printf("\033[H\033[J");
70
71    printf(" %c | %c | %c \n", map[0][0], map[0][1], map
          [0][2]);
72    printf("___|___|___\n");
73    printf("   |   |   \n");
74    printf(" %c | %c | %c \n", map[1][0], map[1][1], map
          [1][2]);
75    printf("___|___|___\n");
76    printf("   |   |   \n");
77    printf(" %c | %c | %c \n", map[2][0], map[2][1], map
          [2][2]);
78    return;
79  }
```

Licenza e crediti

Licenza beerware[1]

Quest'opera è stata redatta da Stefano Cherubin. Mantenendo questa nota, puoi fare quello che vuoi con quest'opera. Se ci dovessimo incontrare e tu ritenessi che quest'opera lo valga, in cambio puoi offrirmi una birra.

[1] http://people.freebsd.org/~phk/