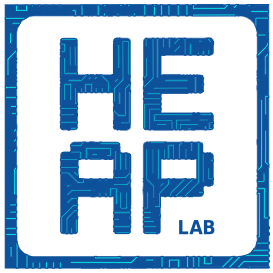
Mastermind in C

Stefano Cherubin*



21/12/2017

[**Informatica A**] Esercitazione #16

corso per Ing. Gestionale a.a. 2017/18

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

Indice

1 Mastermind

Si implementino le seguenti versioni semplificate del famoso gioco crittografico mastermind.

1.1 Versione base

Il gioco si svolge utente contro computer. Il computer genera casualmente un codice come disposizione con ripetizione di lunghezza N. Ciascuna cifra $elem$ del codice (detta anche colore) deve essere rappresentata da un numero intero $elem \in [0; maxColors - 1]$.

La variabile $maxColors$ rappresenta la difficoltà del gioco ed è inserita dal giocatore a inizio partita (difficoltà minima $maxColors = 2$). La lunghezza N del codice (in numero di cifre per codice) viene impostata nel codice tramite `#define N 4`.

Lo scopo del gioco è indovinare il codice generato dal computer nel minor numero di tentativi.

Il turno si svolge nel seguente modo: il giocatore prova ad inserire un codice, il computer analizza il codice inserito e fornisce come indicazione il numero e la posizione degli elementi corretti stampando a video un carattere `const char miss = '_';` in corrispondenza di ogni elemento non indovinato, `const char hit = 'X';` in corrispondenza di un elemento indovinato.

Si stampi a video il numero di turni che il giocatore ha impiegato a indovinare il codice.

1.2 Variante: lunghezza dinamica

Si consideri il gioco descritto al punto precedente. Si renda possibile al giocatore scegliere il numero N di cifre con cui giocare. Il giocatore sceglierà il numero di cifre inserendolo in input dopo aver scelto il numero dei colori. Sia $N = numDigits = 2$ il numero minimo di cifre consentite.

1.3 Variante: difficile come il gioco originale

Si consideri il gioco descritto al punto precedente. Si applichino modifiche alle regole del turno e alle condizioni di fine del gioco.

1.3.1 Regole del turno

Durante ogni turno, l'analisi del computer non indica posizione e numero degli elementi indovinati ma fornisce soltanto due tipi di simbolo in output `const char black = 'X';` e `const char white = 'O';`

black se compare un simbolo black significa che nel codice inserito dall'utente vi era una corrispondenza per colore e posizione rispetto al codice da indovinare. Il numero di simboli black indica il numero di corrispondenze esatte individuate, la loro posizione non ha alcun significato.

white se compare un simbolo white significa che nel codice inserito dall'utente vi era una corrispondenza per colore ma non per posizione rispetto al codice da indovinare. Il numero di simboli white indica il numero di corrispondenze esatte individuate, la loro posizione non ha alcun significato.

1.3.2 Condizione di sconfitta

Si definisca `#define MAXTURN` inoltre un numero massimo di turni che l'utente ha per indovinare. Se l'utente supera questo numero, si dice che il gioco viene vinto dal computer.

1.4 Soluzioni

1.4.1 Versione base

Listato 1: Mastermind (ultra-simplified)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #define LEN 4
5
6  int main() {
7    int sol[LEN];
8    int input[LEN];
9    int i, maxColors, guessed, turn;
10   const char hit = 'X';
11   const char miss = '_';
12
13   /* set random seed to current time */
14   srand(time(NULL));
15
16   printf("*****Welcome to Mastermind*****\n");
17
18   /* set difficulty */
19   printf("Please tell me how many color you'd like to
        play with: ");
20   scanf("%d", &maxColors);
21   while(maxColors <= 1) {
22     printf("\nProvide a number of color greater than 1
          ");
23     scanf("%d", &maxColors);
24   }
25
26   /* compute secret code */
27   for (i = 0; i < LEN; i++) {
28     sol[i] = rand() % maxColors;
29   }
30
31   printf("*** hey, try to guess! ***\n");
32   turn = 0;
33   do {
34     /* input */
35     printf("\nGuess my %d numbers (from 0 to %d)\n",
          LEN, maxColors - 1);
36     for (i = 0; i < LEN; i++) {
37       scanf(" %d", &input[i]);
38     }
```

```
39
40      /* check */
41      guessed = 1;
42      for (i = 0; i < LEN; i++) {
43        if (input[i] == sol[i]) {
44          printf("%c ", hit);
45        } else {
46          printf("%c ", miss);
47          guessed = 0;
48        }
49      }
50      ++turn;
51    } while(!guessed);
52
53    printf("\nCongrats! You've guessed in %d turns!\n",
          turn);
54    return 0;
55  }
```

1.4.2 Lunghezza dinamica

Listato 2: Mastermind (dynamic lenght)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  int main() {
6    int* sol;
7    int* input;
8    int i, maxColors, numDigits, guessed, turn;
9    const char hit = 'X';
10   const char miss = '_';
11
12   srand(time(NULL));
13
14   printf("*****Welcome to Mastermind*****\n");
15
16   /* set difficulty */
17   printf("Please tell me how many color you'd like to
        play with: ");
18   scanf("%d", &maxColors);
19   while(maxColors <= 1) {
20     printf("\nProvide a number of color greater than 1
         ");
21     scanf("%d", &maxColors);
22   }
23
24   printf("Please tell me how many digits should have
        your challenge: ");
25   scanf("%d", &numDigits);
26   while(numDigits <= 1) {
27     printf("\nProvide a number of digits greater than
         1 ");
28     scanf("%d", &numDigits);
29   }
30
31   sol = (int*) malloc(sizeof(int) * numDigits);
32   if (!sol) {
33     printf("Memory Error: can't allocate.\n");
34     return -1;
35   }
36
37   input = (int*) malloc(sizeof(int) * numDigits);
38   if (!input) {
```

```
39      free(sol);
40      printf("Memory Error: can't allocate.\n");
41      return -1;
42    }
43
44    for (i = 0; i < numDigits; i++) {
45      sol[i] = rand() % maxColors;
46    }
47
48    printf("*** hey, try to guess! ***\n");
49    turn  = 0;
50    do {
51      printf("\nGuess my %d numbers (from 0 to %d)\n",
           numDigits, maxColors - 1);
52      for (i = 0; i < numDigits; i++) {
53        scanf(" %d", &input[i]);
54      }
55
56      guessed = 1;
57      for (i = 0; i < numDigits; i++) {
58        if (input[i] == sol[i]) {
59          printf("%c ", hit);
60        } else {
61          printf("%c ", miss);
62          guessed = 0;
63        }
64      }
65      ++turn;
66    } while(!guessed);
67
68    printf("\nCongrats! You've guessed in %d turns!\n",
         turn);
69    free(input);
70    free(sol);
71    return 0;
72  }
```

1.4.3 Variante difficile come l'originale

Listato 3: Mastermind (like the original)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  #define MAXTURNS 10
6
7  int main() {
8    int* sol;
9    int* input;
10   int i, j, maxColors, numDigits, guessed;
11   int turn, found;
12   const char black = 'X';
13   const int BLACK_INPUT = -1;
14   const char white = 'O';
15   const int WHITE_INPUT = -2;
16
17   srand(time(NULL));
18
19   printf("*****Welcome to Mastermind*****\n");
20   printf("Please tell me how many color you'd like to
        play with: ");
21   scanf("%d", &maxColors);
22   while(maxColors <= 1) {
23     printf("\nProvide a number of color greater than 1
          ");
24     scanf("%d", &maxColors);
25   }
26
27   printf("Please tell me how many digits should have
        your challenge: ");
28   scanf("%d", &numDigits);
29   while(numDigits <= 1) {
30     printf("\nProvide a number of digits greater than
         1 ");
31     scanf("%d", &numDigits);
32   }
33
34   sol = (int*) malloc(sizeof(int) * numDigits);
35   if (!sol) {
36     printf("Memory Error: can't allocate.\n");
37     return -1;
38   }
```

```
39
40    input = (int*) malloc(sizeof(int) * numDigits);
41    if (!input) {
42      free(sol);
43      printf("Memory Error: can't allocate.\n");
44      return -1;
45    }
46
47    for (i = 0; i < numDigits; i++) {
48      sol[i] = rand() % maxColors;
49    }
50
51    printf("*** hey, try to guess! ***\n");
52    turn  = 0;
53    do {
54      printf("\nGuess my %d numbers (from 0 to %d)\n",
          numDigits, maxColors - 1);
55      for (i = 0; i < numDigits; i++) {
56        scanf(" %d", &input[i]);
57      }
58
59      /* look for blacks */
60      guessed = 1;
61      for (i = 0; i < numDigits; i++) {
62        if (input[i] == sol[i]) {
63          input[i] = BLACK_INPUT;
64        } else {
65          guessed = 0;
66        }
67      }
68
69      /* look for whites */
70      for (i = 0; i < numDigits; i++) {
71        if (input[i] != BLACK_INPUT) {
72          found = 0;
73          for (j = 0; j < numDigits && !found; j++) {
74            if (input[j] == sol[i]) {
75              input[j] = WHITE_INPUT;
76              found = 1;
77            }
78          }
79        }
80      }
81
82      /* output */
83      for (i = 0; i < numDigits; i++) {
```

```
84        if (input[i] == BLACK_INPUT) {
85          printf("%c ", black);
86        } else if (input[i] == WHITE_INPUT) {
87          printf("%c ", white);
88        }
89      }
90
91      ++turn;
92    } while(!guessed && turn < MAXTURNS);
93
94    if (turn < MAXTURNS) {
95      printf("\nCongrats! You've guessed in %d turns!\n"
            , turn);
96    } else {
97      printf("\nComputer Wins\n");
98    }
99
100   free(input);
101   free(sol);
102   return 0;
103 }
```

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/