Laurea In Ingegneria dell'Informazione

Esercitazioni Guidate di Tecniche della Programmazione

Note introduttive: Seguire le raccomandazioni date nelle precedenti EG …

Esercitazione "Per le vacanze"

EGV 1 Il gioco dell'11, no, del 15, no, del 21, e altri giochi

Tanti anni fa qualcuno mi spiegò il *gioco degli 11 fiammiferi* …

- Ci sono 11 fiammiferi sul tavolo, e due giocatori (intorno al tavolo).
- Ogni giocatore, quando è il suo turno, deve prelevare almeno un fiammifero. E puo' prenderne fino a 3.
- Perde chi prende l'ultimo fiammifero. Oppure … Vince chi lascia l'avversario con un solo fiammifero sul tavolo.

In questo esercizio scriveremo programmi per far giocare una persona (Human, person, unità a carbonio, essere umano) contro una macchina (Machine) …

Prima di scrivere codice però, è bene cercare di capire quale strategia vincente si può codificare nel programma, per farlo vincere (cioè per far vincere la macchina).

Esegui, o eseguite, almeno 10-15 partite, giocatore1 contro giocatore2. Cercate di capire come un giocatore può decidere quanti fiammiferi prendere quando è il suo turno. Cercate anche di capire se iniziare per primi procura un vantaggio.

Suggerimenti seguono …

Un primo programma potrebbe essere ispirato dall'output qui sotto: il programma è in `fiammiferi1.c` ma non andate subito a guardare. Piuttosto, in questo e nel prossimo suggerimento cercate di capire come la macchina (che parte per prima) decide le proprie mosse. Ci sono delle regolarità?

```
Oh sfidante, comincio io ...

-- la mia mossa e` prendere 2 fiammiferi
ora sul tavolo ci sono 9 fiammiferi

-- la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 2
ora sul tavolo ci sono 7 fiammiferi

-- la mia mossa e` prendere 2 fiammiferi
ora sul tavolo ci sono 5 fiammiferi

-- la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 1
ora sul tavolo ci sono 4 fiammiferi

-- la mia mossa e` prendere 3 fiammiferi
ora sul tavolo ci sono 1 fiammiferi

-- la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 1
ora sul tavolo ci sono 0 fiammiferi


    AH AH, hai perso!
FINE
```

Si nota, forse, che la macchina sceglie la propria mossa in base alla mossa del giocatore (tranne la prima).

Scrivere l'algoritmo e il relativo programma.
Un suggerimento per l'algoritmo segue …

Il programma dovrà realizzare un algoritmo di questo tipo

0) Intanto definiamo delle grandezze costanti per il numero di fiammiferi e per la dimensione della presa (3 nel caso descritto sopra); poi servono variabili per rappresentare il numero di fiammiferi presenti sul tavolo (nFiamm, che cambia durante il gioco), la mossa del giocatore (mossaG) e quella della macchina (mossaM). La mossa e' il numero di fiammiferi che si preleva.
1) Comincia la macchina. E prende 2 fiammiferi. mossaM=2
2) Ripeti
    a. Aggiorna nFiamm (nFiamm = nFiamm – mossaM)
    b. Mostra nFiamm all'utente
    c. Chiedi mossaG
    d. Aggiorna NFiamm
    e. Mostra nFiamm
    f. Calcola la prossima mossaM: mossaM = qualcosa in relazione all'ultima mossaG

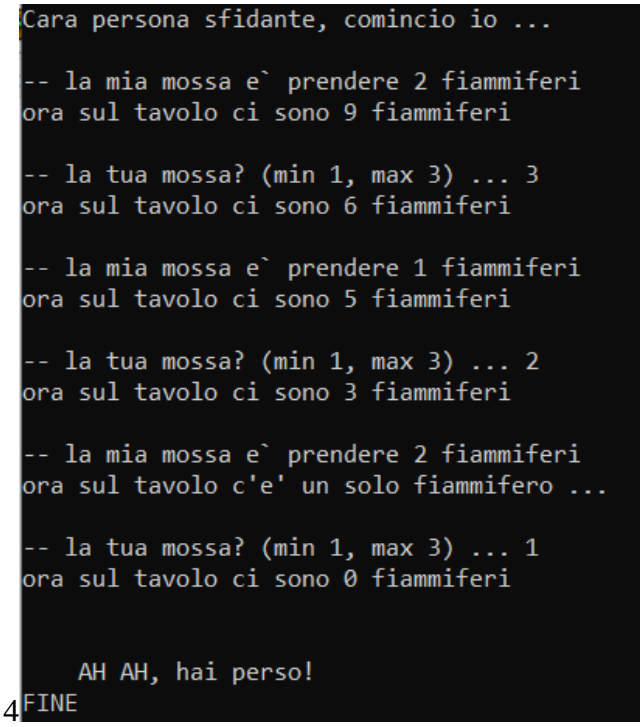
   Mentre nFiamm>0
3) Stampare "hai perso"
4) fine

Ultimo suggerimento segue

Qui c'e' proprio la soluzione … ma non una spiegazione del perché funziona: http://tecnomaux.weebly.com/11-fiammiferi.html

Chiamate il file che contiene questo programma `mioFiammiferi1.c`
O come vi pare …
Quando avete fatto e avete testato il programma giocando ci varie volte … proseguite

In effetti scrivere "ci sono 1 fiammiferi" è brutto.
Si può correggere e ottenere un programma come in `fiammiferi2.c` del quale si vede l'output qui sotto.



4

Se lo avete già fatto proseguite …
Sennò fatelo, nel file `mioFiammiferi2.c` e proseguite.

4.28. Miglioriamo il programma …

Nei due programmi precedenti, se siete stati disattenti come me, l'inserimento di input illegali non viene gestito bene e provoca problemi (provate a inserire una mossa troppo grande, o troppo piccola, o a dire 2 quando dovreste prendere l'ultimo fiammifero (e quindi dovreste invece scrivere 1) … Correggere prego … (Poi guardate `fiammiferi3.C`, del quale l'output di una esecuzione segue).

```
Cara persona, comincio io ...

-- la mia mossa e` prendere 2 fiammiferi
ora sul tavolo ci sono 9 fiammiferi

-- la tua mossa? (min 1, max 3) ... 4
... no, scelta illegale: cambiamo la scelta in 1
ora sul tavolo ci sono 8 fiammiferi

-- la mia mossa e` prendere 3 fiammiferi
ora sul tavolo ci sono 5 fiammiferi

-- la tua mossa? (min 1, max 3) ... -2
... no, scelta illegale: cambiamo la scelta in 1
ora sul tavolo ci sono 4 fiammiferi

-- la mia mossa e` prendere 3 fiammiferi
ora sul tavolo c'e' un solo fiammifero ...

-- la tua mossa? (min 1, max 3) ... -2
... no, scelta illegale: cambiamo la scelta in 1
ora sul tavolo ci sono 0 fiammiferi


    AH AH, hai perso!
FINE
```

Disquisizione … Poi ne riparliamo

Ok, questo è un gioco "deterministico"; in particolare, chi comincia per primo e non commette errori vince.

Quindi se programmiamo le mosse come visto sopra (mossa della macchina = 4 – mossa del giocatore) e facciamo cominciare la macchina … la macchina vince.

Se comincia l'umano, e non commette errori, vince l'umano.

Guardate qualche formalizzazione e spiegazione relativa al gioco e a giochi consimili …

- Qui (https://mathbox.latteseditori.it/matematica-ricreativa/vuoi-un-fiammifero) c'è una disamina delle caratteristiche del gioco, da cui si capisce che si possono ottenere tante varianti, con numero iniziale di fiammiferi e dimensione della presa diverse.
  (15 fiammiferi, con prese fino a 4, o 5 … 21 fiammiferi con prese da 4 o 5 o 6 … etc.)

- Qui (http://utenti.quipo.it/base5/jsmarienbad/jsmarienbad.htm) c'è un esame di un gioco simile (ma diverso) che a sua volta è basato su un altro gioco (Nim) … chissà, magari vi piace fare esercizi anche su questi giochi.
  Si tratta di altri giochi deterministici in cui ci si imbatte quando si cerca qualcosa sul gioco degli 11, o 15, fiammiferi.

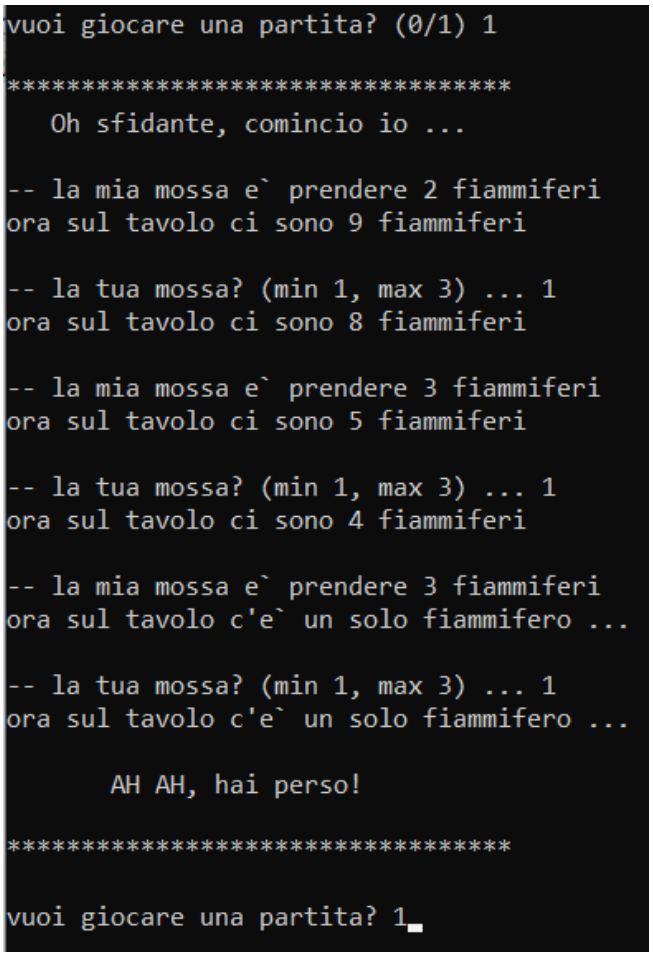
Dovreste riuscire a capire come, in base al numero iniziale di fiammiferi (11, 15, 21, 88 …) e alla dimensione massima della presa (3, 4, 5, …) si possa determinare una sequenza di "configurazioni perdenti" …
… in realta' solo in base alla dimensione massima della presa …

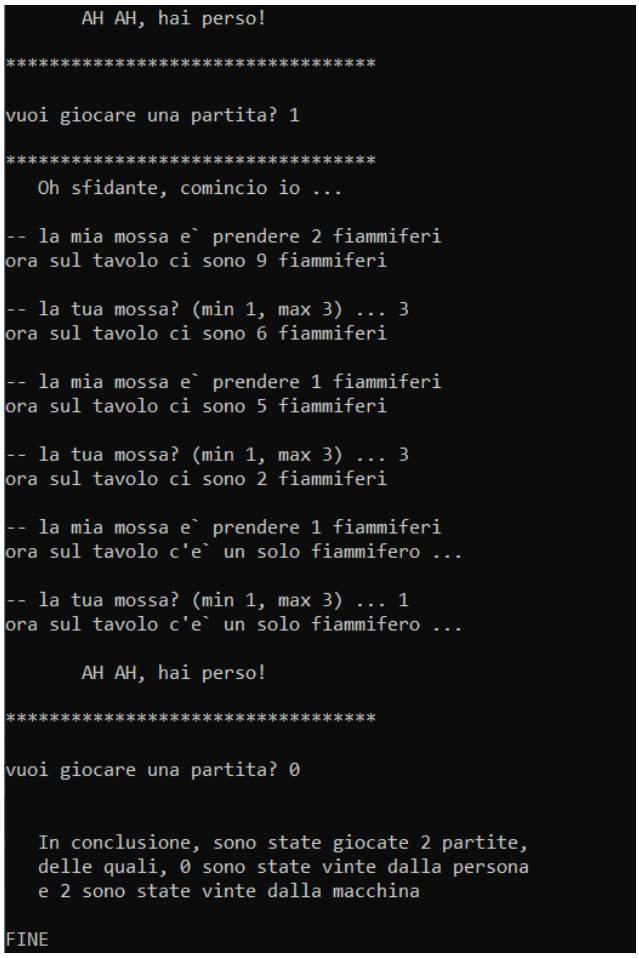
Una configurazione perdente è un numero di fiammiferi sul tavolo per il quale si può dire che *"chi riceve quella configurazione e su quella deve fare la sua mossa, perderà … a meno che l'avversario non faccia errori"*.

Fine della disquisizione

4.29. Rilassiamoci per migliorare ancora di più

Dopo averci un po' pensato, rilassatevi modificando `fiammiferi3.c` (anzi … `mioFiammiferi3.c`, se avete chiamato così il vostro programma) in modo che permetta di svolgere più partite nella stessa esecuzione del programma, tenendo il conto di quante ne ha vinte il computer e quante la persona.





In queste modifiche useremo delle funzioni:

- una per ricevere l'input sulla mossa della persona;
- una per stampare il numero di fiammiferi attualmente sul tavolo;

  queste due funzioni ci consentono di avere una main() piu' pulita ...
- e una per giocare una partita.

Prima scrivete l'algoritmo:
le variabili relative alla partita saranno nella funzione che la gioca, chiamata `unaPartita()`;
la `main()` avrà bisogno di variabili per far scegliere se continuare a fare partite, e per contenere il punteggio delle vittorie di macchina e persona … anche se, nella situazione attuale, la persona è condannata …

segue un suggerimento su `unaPartita()`

Suggerimento

`unaPartita()` potrebbe avere come parametri il numero iniziale di fiammiferi e la dimensione della presa: non è una cosa indispensabile adesso, dato che questi valori sono più o meno fissati nel programma.
Ma questa organizzazione verrà buona quando dovremo generalizzare il programma a qualsiasi gioco, in cui il numero iniziale dei fiammiferi e la presa massima siano diversi.

La possibilità di far cominciare la persona … verrà considerata in un altro momento.

Una soluzione è in `fiammiferi4.c`
Ma non volete guardarla subito.

Suggerimento sull'algoritmo segue

[
Osservazione: anticipando quel che faremo dopo, notiamo che se vogliamo usare questo programma per altri giochi dei fiammiferi, con diversi numeri iniziali di fiammiferi, in molti casi – ma non tutti – il programma funziona ancora bene … potete provare e vedere in quali casi non funziona … se avete letto bene le pagine linkate sopra sapete già perché';
se no, facendo queste prove, e pensando a come farle, sarà più facile capire cosa sono le configurazioni perdenti.
Tutto questo non è indispensabile adesso.
]

Suggerimento

Il programma principale in sostanza chiede se si vuole giocare e tiene traccia dei risultati delle partite …

0) nFiamm, dimPresa come prima,
   score sarebbe il risultato di una partita,
   scoreG, scoreM le partite vinte da G ed M …
   nPartite il numero di partite giocate
   - La funzione unaPartita() riceve come parametri il numero iniziale di fiammiferi e la dimensione massima della presa, gioca una partita e poi restituisce 1 se ha vinto la persona, 0 se ha vinto la macchina …

1) Chiedere e leggere se si vuole fare una partita (var scelta)

2) Ripetere, mentre scelta e' divers o da zero
   a. score = unaPartita(nFiamm, dimPresa);
   b. incrementare nPartite
   c. incrementare scoreG o scoreM a seconda del valore di score
   d. chiedere e leggere in scelta se si vuole giocare ancora

3) stampare i risultati

4) fine

Scrivere ora il programma, includendo il prototipo per `unaPartita()`

Poi scrivete, basandovi sull'uso che ne facciamo nel programma, la definizione di `unaPartita()`

Riguardo a `unaPartita()`segue un suggerimento sull'algoritmo.

Suggerimento

La funzione `unaPartita()` riceve il numero iniziale di fiammiferi e la dimensione massima della presa e gioca una partita.

L'algoritmo che la guida è molto simile al primo visto in questo esercizio (fiammiferi1)

0) fiammiferi, presa
1) Comincia la macchina. E prende 2 fiammiferi. mossaM=2
2) **Ripeti**
    a. Aggiorna fiammiferi

    b. Stampa dei fiammiferi ora disponibili:
    questo è un sottoproblema, risolto con una chiamata di funzione …
    `stampaFiammiferi(fiammiferi)`
    questa e' una funzione che stampa in output il numero di fiammiferi ricevuti. Lo fa usando bene le parole "fiammiferi" e "fiammifero" come visto sopra;

    c. Chiedi mossaG.
    Anche questo è un sottoproblema: si usa una funzione che restituisce la mossa dichiarata, se legittima, oppure 1.
    Per sapere se una mossa è legittima la funzione ha bisogno di conoscere tre informazioni cruciali: il valore della presa minima, quello della presa massima, e anche minimo, il numero attuale di fiammiferi.

    d. Aggiorna fiammiferi

    e. stampaFiammiferi(fiammiferi)

    f. Calcola la prossima mossaM: mossaM = 4-…

    Mentre fiammiferi>0

3) Stampare "hai perso"
4) fine

Scrivere la definizione della funzione.

Scrivere i prototipi per

```
stampaFiammiferi()
letturaMossPersona()
```

E poi scrivere gli algoritmi relativi

Suggerimenti sulle intestazioni delle funzioni seguono

Suggerimento

I dua valori

- numero iniziale dei fiammiferi
- dimensione massima della presa (assumendo che la minima e' sempre 1)

sono in realta' vere e proprie caratteristiche fondamentali del gioco:

"11 fiammiferi con prese da 3" e *"15 fiammiferi con prese da 4"*,

sono giochi in cui la strategia di gioco è diversa.

Dovreste gia' sapere perché' … ma lo rivedremo.

Per ora però accontentiamoci di avere queste due caratteristiche come parametri della partita … così sarà più semplice generalizzare (quando generalizzeremo).

```
int unaPartita(int fiammiferi, int presa);
```
riceve le caratteristiche fondamentali del gioco, cioè il numero iniziale di fiammiferi e la dimensione massima della presa (quella minima è sempre 1) e gioca una partita come visto nelle pagine precedenti.

Suggerimenti sulle altre funzioni seguono

Suggerimento

```
int letturaMossaPersona(int, int, int);
```
i parametri sono la mossa minima (in questo esercizio finora sempre 1), la presa massima (in questo esercizio finora sempre 3) e il numero attuale di fiammiferi sul tavolo.
Il terzo serve perche' se la mossa cercasse di superarlo non sarebbe legittima.

Suggerimento

```
void stampaFiammiferi(int );
```
riceve il numero attale di fiammiferi sul tavolo e lo stampa correttamente

Quando il programma e' fatto e testato, proseguite.

4.30. Dopo il quarto programma … bisogna fare uno sforzo di astrazione

Dopo il quarto programma … bisogna fare uno sforzo di astrazione

E capire bene quali strumenti abbiamo per generalizzare il programma
al caso di una qualsiasi coppia

‹N, D›

dove N = numero iniziale di fiammiferi
e D = dimensione max delle prese (dim. minima assumiamo sempre 1)

Dalla documentazione proposta prima, si dovrebbe evincere un fatto fondamentale:

dato D è possibile determinare una sequenza di "configurazioni perdenti", che inizia con 1 e prosegue con numeri ottenuti dal precedente mediante l'aggiunta di D+1.

Ad esempio, per D=3 la sequenza delle configurazioni perdenti è ottenuta a partire da 1 aggiungendo ogni volta 4 …

[1, 5, 9, 13, 17, 21, …]

Il che ci dice anche che se fosse il "gioco dei 21 fiammiferi", on presa massima 3, chi iniziasse partirebbe da una configurazione perdente … (cioè perderebbe sempre, a meno di errori dell'avversario).

Se la configurazione iniziale (N fiammiferi sul tavolo) "non è perdente", chi ha il primo turno vince sempre, a meno di imbrogliarsi da solo facendo un errore (presa sbagliata).

È per questo che il programma, nel gioco degli undici fiammiferi vuole sempre partire per primo … ma lo avevate capito …

La strategia vincente, a patto di iniziare per primi, è di fare in modo che quando tocca all'avversario, questi abbia sul tavolo una configurazione perdente.

Qualunque mossa M faccia l'avversario, se la successiva mossa (nostra, o della macchina) è 4-M, ricacciamo l'avversario in una configurazione perdente.
Questo nel caso degli 11 fiammiferi, ma anche dei 15, o 22, o 23, o 24 … provvisto che D sia 3 (21 no … e' perdente …).

Quali sono le configurazioni perdenti, dato N, se D = 4? E se D = 5?

Risposta più avanti … datela adesso però …

Caso D=4; configurazioni perdenti: [1, 6, 11, 16, 21, 26, 31, …]

Caso D=5; configurazioni perdenti: [1, 7, 13, 19, 25, 31, 37, …]

OK, adesso l'esercizio finale è:
fare un programma in cui, assegnati come costanti N e D, sia possibile eseguire una serie di partite al *gioco degli N fiammiferi* con *presa massima D*,
ottenendo alla fine la classifica

Numero di vittorie della macchina
Numero di vittorie della persona

La *macchina* ha la prima mossa, come nei programmi precedenti.
(Dopo, chi vuole può aggiungere la possibilità di far decidere all'utente chi inizia prima – ma non ora).

Il programma deve

- Procedere assumendo che N non sia una configurazione perdente …
- usare un array di 40 elementi interi, riempito con le prime 40 configurazioni perdenti relative a D (vedi la nota successiva …)
- durante una partita, per ogni mossa della macchina, determinare la mossa in modo da far raggiungere ai fiammiferi sul tavolo la prossima configurazione perdente (che viene così servita all'avversario umano).

Per fare questo si scandisce l'array, cercando il valore massimo nell'array, che sia più piccolo del numero attuale di fiammiferi; chiamiamo `proxConf` questo elemento dell'array: allora la mossa sara' calcolata in modo da far rimanere sul tavolo `proxConf` fiammiferi …

(Per scandire l'array, usare una funzione chiamata `maxMinoreDi()`, che
Ricevendo un array di interi, `arr`, e un valore intero, `val`
restituisca il valore massimo nell'array che è anche minore di `val`, oppure `-1` se un tale valore non c'è.

NB
40 è un numero di configurazioni perdenti che assumiamo abbastanza grande da non darci problemi …
In particolare, se usassimo fino a 155 fiammiferi, con presa massima uguale a 3, saremmo sempre sicuri di trovare nell'array la configurazione perdente più vicina al numero di fiammiferi attualmente sul tavolo …
Per numeri piu' grandi questa sicurezza non c'è e il programma va corretto (con un array più grande …)
Ma chi vuole usare più di 155 fiammiferi???
Davvero … anche con prese da 4, 5, 6 se N e' piu' di 30-40 … la partita diventa troppo lunga e ripetitiva … meglio limitarsi …

L'algoritmo del programma è lo stesso di prima …
L'unica differenza è che ora c'è un array (`configurazioniPerdenti`) da inizializzare: per inizializzarlo usare una funzione
`inizializzaArray()` che riceve l'array e il valore della presa massima, e riempie l'array.

Scrivere l'algoritmo della funzione che modifica `unaPartita()` in modo da renderla capace di giocare partite di qualsiasi tipo, dati il numero iniziale di fiammiferi, la dimensione della presa massima e l'array con le configurazioni perdenti (di dimensione nota 40).
Chiamare questa funzione `unaPartitaQualsiasi()`

Poi ottenere il programma complessivo.
Questa è un'estensione di `fiammiferi4.c` molto più attraente …
(io la chiamo `fiammiferi5.c` - chiamatela `mioFiammiferi5.c` …)

Suggerimenti sulle funzioni seguono

Suggerimento sul'algoritmo di `unaPartitaQualsiasi()`

e' molto simile a quello visto per `unaPartita()`

Qui pero', quando si deve definire la prossima mossa (MossaM) della macchina,

- intanto non iniziamo con la prima mossa che prende 2 fiammiferi … qui dobbiamo lavorare qualsiasi siano i valori di N e D (2 andava bene per 11 fiammiferi e prese al massimo di 3 … in quel modo portavamo l'avversario a 9 fiammiferi (cioe' nella piu' vicina configurazione perdente)
- e poi il calcolo si basa
  - sulla determinazione del valore della configurazione perdente piu' vicna cui possiamo arrivare togliendo tra 1 e D fiammiferi (proxConf)
  - sul numero attuale di fiammiferi sul tavolo (all'inizio N … poi di meno, conservato in una variabile apposita, ad esempio actualFiamm)

Quanti fiammiferi bisogna togliere da actualFiamm per portare il numero dei fiammiferi a proxConf? Quella e' la mossaM, in qualsiasi turno della macchina, anche il primo..

Ci sara' un altro suggerimento su questa funzione alla fine, se serve.

Suggerimento su `inizializzaArray()` segue

Suggerimento su `inizializzaArray()`

`void inizializzaArray(int arr[DIM], int);`
in fiammiferi5.c, ad esempio, riceve, con la chiamata
inizializzaArray(configurazioniPerdenti,dimPresa);
l'array che deve essere inizializzato con le configurazioni perdenti e il valore della presa massima

Segue un suggerimento su come definire la configurazione perdente i-esima

Suggerimento

La prima configurazione perdente e' sempre 1 …

La i-esima e' `arr[i] = arr[i-1] + d + 1;`
dove `arr` e' l'array da inizializzare e `d` è la dimensione massima della presa

segue un suggerimento sulla funzione `maxMinoreDi()`

Suggerimento

```
int maxMinoreDi(int arr[DIM], int val) {
```

si tratta di scandire l'array, quindi serve un classico contatore `i`

al crescere di `i`,
se i e' troppo grande (cioe' ha raggiunto il limite dell'array … non ci sono piu' elementi dell'array da analizzare

se stiamo analizzando `arr[i]` dobbiamo verificare se `arr[i]` e' maggiore del numero attuale di fiammiferi

quindi questa funzione deve ricevere sia l'array che il numero attuale di fiammiferi sul tavolo, con la chiamata `maxMinoreDi(confPerdenti, actualFiamm);`

In realta', quando effettuiamo questa chiamata la macchina sta determinando la sua prossima mossa;
siamo nella funzione `unaPartitaQualsiasi()`
questo numero e' la prossima configurazione perdente da servire all'avversario …
Quindi effettivamente la chiamata andrebbe fatta per assegnare ad una variabile questa prossima configurazione:

```
proxConf = maxMinoreDi(confPerdenti, actualFiamm);
```

Cosa si fa con `proxComp` viene accennato nel prossimo suggerimento, che riguarda la funzione `unaPartitaQualsiasi()`

Ultimo Suggerimento

```
int unaPartitaQualsiasi(int nInitFiam, int presaMax, int confPerdenti[DIM]);
```

fa giocare una partita e restituisce 0 o 1 a seconda di chi ha vinto

viene chiamata dalla main()

inizia la macchina

riceve

- il numero iniziale di fiammiferi stabilito per il tipo di gioco usato dalla main()
- la dimensione della presa massima
- (questi due valori sono le caratteristiche del tipo di gioco dei fiammiferi che si gioca nel programma, e sono stabilite da due costanti
- La funzione riceve anche l'array delle configurazioni perdenti, riempito per bene in ordine crescente a partire da 1

Una variabile `proxConf` viene assegnata con la prossima configurazione perdente da servire al malcapitato avversario.

Si può calcolare la prossima mossa come Numero attuale di fiammiferi – proxConf

Fiammiferi5.c contiene una implementazione del programma qui descritto;
quando avete sviluppato la vostra soluzione confrontatela con questa.
Oppure cerca lì ispirazione …

Quel file contiene anche qualche indicazione per eventuali azioni di debugging, corrispondenti a situazioni prevedibili di errore …