# Laurea In Ingegneria dell'Informazione

# Esercitazioni Guidate di Tecniche della Programmazione

**Note introduttive:**

1) Nel titolo di ogni sezione di questo documento è specificato tra parentesi il nome del (o dei) file in cui è proposta una soluzione (se disponibile nella directory "`programmi`" di questa EG).
2) I programmi che scriveremo dovranno essere in accordo con la definizione standard **ANSI C** del linguaggio C.
    a. Se usate un sistema diverso dal DEV, provvedete a che la compilazione avvenga con il compilatore standard C.
    b. Ricordate che un programma C e' in un file con estensione "`.c`"
    c. Se usate il DEVC++, per configurare bene la compilazione bisogna
        i. andare nel menù "Tools", selezionare "Compiler Options", scegliere "Settings" e poi "C Compiler"; poi selezionare almeno "Support all ANSI Standard C Programs")
        ii. (se l'interfaccia è in italiano …) andare nel menu' "Strumenti", selezionare "Opzioni di compilazione", "Compilatore", "Generazione di Codice …", "Compilatore C" e poi far apparire "Yes" almeno accanto a "Supporto programmi ANSI standard C.
    d. NB le immagini in queste dispense sono prese dalla versione 4.9.9.2 del DEV. La versione 5.11 è probabilmente la più recente e può differire solo in qualche dettaglio.

## Esercitazione "Per le vacanze invernali"

### *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 può 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, persona, unità a carbonio …) 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, anche meglio, eseguite almeno una dozzina di 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.

Di seguito un'immagine di come potrebbe essere il programma che realizza una partita.

(Vedi anche dopo l'immagine, subito).

```
[Sul tavolo ci sono 11 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 2 fiammiferi

[Sul tavolo ci sono 9 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 2
MACHINE **** Ok, hai scelto di prendere 2 fiammiferi

[Sul tavolo ci sono 7 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 2 fiammiferi

[Sul tavolo ci sono 5 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 1
MACHINE **** Ok, hai scelto di prendere 1 fiammiferi

[Sul tavolo ci sono 4 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 3 fiammiferi

[Sul tavolo ci sono 1 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 1
MACHINE **** Ok, hai scelto di prendere 1 fiammiferi

MACHINE **** Bip ... Bip ... Oh Human, hai preso l'ultimo fiammifero ... Bip ...
MACHINE **** Bip ... Bip ... Ho vinto :) ... Bip ...
Grazie della partita :)
```

Dopo aver fatto la dozzina di partite (e in due di solito si ragiona meglio) potreste intanto provare a scrivere il programma che fornisca l'output qui sopra.

Per farlo, prima scrivete l'algoritmo in linguaggio naturale e pseudocodice.
Cosa c'è nel passo 0) ? (Che si può completare strada facendo, al solito)
Quali passi si eseguono, a parte stampare il dialogo e le informazioni su quanti fiammiferi sono correntemente sul tavolo?

- Calcolare ed applicare la prima mossa (che per definizione spetta alla macchina, sennò finisce la magia)
- Ricevere ed applicare la mossa della persona
- Calcolare ed applicare la mossa della macchina

(e queste ultime due operazioni si eseguono ripetutamente … mentre ci sono fiammiferi sul tavolo)

Se non è chiaro come calcolare la mossa della macchina, scegliete di farle prendere un fiammifero. Questo ovviamente **non è il metodo per vincere**, ma così il programma riesce a funzionare … sul calcolo esatto della mossa si rifletterà dopo, se non è chiaro.

Un programma-soluzione è in `fiammiferi0.C` ma non andate subito a guardare.
Prima l'algoritmo.
Poi, guardate l'algoritmo … incompleto … proposto nella prossima pagina.

Nel frattempo, in base agli esperimenti di gioco che avete fatto, cercate di capire come la macchina (che parte per prima) decide le proprie mosse. Ci sono delle regolarità?
! Come sarebbe non avete preso nota delle mosse fatte nelle partite sperimentali??
Rifatele e prendete nota!

**Poi** segue un suggerimento.

Qui ci sono due ulteriori esecuzioni del programma

```
[Sul tavolo ci sono 11 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 2 fiammiferi

[Sul tavolo ci sono 9 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 3
MACHINE **** Ok, hai scelto di prendere 3 fiammiferi

[Sul tavolo ci sono 6 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 1 fiammiferi

[Sul tavolo ci sono 5 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 1
MACHINE **** Ok, hai scelto di prendere 1 fiammiferi

[Sul tavolo ci sono 4 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 3 fiammiferi

[Sul tavolo ci sono 1 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 1
MACHINE **** Ok, hai scelto di prendere 1 fiammiferi

MACHINE **** Bip ... Bip ... Oh Human, hai preso l'ultimo fiammifero ... Bip ...
MACHINE **** Bip ... Bip ... Ho vinto :) ... Bip ...
Grazie della partita :)
```

```
[Sul tavolo ci sono 11 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 2 fiammiferi

[Sul tavolo ci sono 9 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 1
MACHINE **** Ok, hai scelto di prendere 1 fiammiferi

[Sul tavolo ci sono 8 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 3 fiammiferi

[Sul tavolo ci sono 5 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 1
MACHINE **** Ok, hai scelto di prendere 1 fiammiferi

[Sul tavolo ci sono 4 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 3 fiammiferi

[Sul tavolo ci sono 1 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 1
MACHINE **** Ok, hai scelto di prendere 1 fiammiferi

MACHINE **** Bip ... Bip ... Oh Human, hai preso l'ultimo fiammifero ... Bip ...
MACHINE **** Bip ... Bip ... Ho vinto :) ... Bip ...
Grazie della partita :)
```

**Suggerimento**
Si nota, forse, che la macchina sceglie la propria mossa in base alla mossa del giocatore (tranne la prima). E la scelta dipende anche dal fatto che i fiammiferi sono 11, e/o che si possono prendere al più tre fiammiferi.

Un suggerimento per l'algoritmo segue …

**Suggerimento**
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`**) (dopo la mossa nFiamm deve cambiare, per continuare a dire quanti fiammiferi sono disponibili sul tavolo …)
   b. Mostra **`nFiamm`** all'utente (mostrare la situazione attuale sul tavolo)
   c. Chiedi **`mossaG`**
   d. Aggiorna **`nFiamm`** (dopo la mossa nFiamm deve cambiare, per continuare a dire quanti fiammiferi sono disponibili sul tavolo …)
   e. Mostra **`nFiamm`** (mostrare la situazione attuale sul tavolo)
   f. Calcola la prossima mossa della macchina: **`mossaM`** = ☺ qualcosa in relazione all'ultima mossaG

   **Mentre `nFiamm`**!=0
3) Stampare "hai perso"
4) fine

Vabbè, facciamo il programma?

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

😇Non era vero che il programma soluzione si chiama `fiammiferi0.c` … in realtà si chiama `fiammiferi1.c`
La versione 0 è quella corrispondente all'algoritmo suggerito sopra, senza l'istruzione *clou* …

Fatto questo, proseguite

Ok, fatto questo programma, miglioriamolo …

Ad esempio, nella soluzione proposta in `fiammiferi1.c`
"`ci sono 1 fiammiferi`" è brutto. Anche "`prendere 1 fiammiferi`" è brutto.

Si può correggere e ottenere un programma che produca l'output come di seguito.
Chiamate il file **`mioFiammiferi2.c`**

```
Cara persona sfidante, come da accordi comincio io ...

[Sul tavolo ci sono 11 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 2 fiammiferi

[Sul tavolo ci sono 9 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 1
MACHINE **** Ok, hai scelto di prendere 1 fiammifero

[Sul tavolo ci sono 8 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 3 fiammiferi

[Sul tavolo ci sono 5 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 3
MACHINE **** Ok, hai scelto di prendere 3 fiammiferi

[Sul tavolo ci sono 2 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 1 fiammifero

[Ora sul tavolo c'e` un solo fiammifero (eh eh ...)]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 1
MACHINE **** Ok, hai scelto di prendere 1 fiammifero

MACHINE **** Bip ... Bip ... Oh Human, hai preso l'ultimo fiammifero ... Bip ...
MACHINE **** Bip ... Bip ... Ho vinto :) ... Bip ...
Grazie della partita :)
```

Una possibile soluzione è in `fiammiferi2.C`

Un suggerimento su come scrivere "fiammifero" oppure "fiammiferi" segue

**Suggerimento 1 di 2**
Nel programma soluzione c'e` questa printf

```
printf("\n\n[Sul tavolo ci sono %d fiammiferi]\n", nFiamm);
```

Il fatto è che quando `nFiamm` vale 1 bisognerebbe scrivere "fiammifero" … con la 'o'

Quindi basta un controllo:

se nFiamm==1
si fa `printf("\n\n[Sul tavolo ci sono %d fiammiferi]\n", nFiamm);`
senno` nella printf ci va `"\n\n[Sul tavolo c'e` un solo fiammifero]\n"`

Questo però risolve solo una occorrenza del problema …
Ci sono altri punti in cui il problema appare e vanno corretti

Correggili, sempre nel medesimo file .c che stai usando.

Poi guarda il suggerimento seguente.

**Suggerimento 2/2**

Dunque, nel programma c'è

```
printf("MACHINE **** Oh Human, la mia mossa e` prendere %d fiammiferi ", mossaM);
```

si può migliorare la stampa, ottenendo quel che si vede nell'immagine precedente, facendo in modo che venga stampata la sequenza di caratteri

```
"...fiammifer%c"
```

dove il carattere stampato al posto di `%c` è un'espressione che produce il valore carattere `'o'` oppure il valore carattere `'i'`, a seconda del valore di `mossaM`

come si fa?
Vedi in fondo alla pagina

Con un'espressione condizionale, in cui viene valutata la condizione `(mossaM==1)` e prodotto un carattere o l'altro …vedi `fiammiferi2.c`

### *4.28. Miglioriamo di più il programma …*

Nei due programmi precedenti, se siete stati disattenti come me, l'inserimento di input illegali non viene gestito bene e provoca problemi
… se non lo avete fatto … 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 … qui sotto un esempio di output

```
Cara unita` a carbonio, come da accordi comincio io ...

[Sul tavolo ci sono 11 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 2 fiammiferi

[Sul tavolo ci sono 9 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 0
... no, scelta illegale: tsk, cambiamo la scelta in 1
MACHINE **** Ok, hai scelto di prendere 1 fiammifero

[Sul tavolo ci sono 8 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 3 fiammiferi

[Sul tavolo ci sono 5 fiammiferi]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 8
... no, scelta illegale: tsk, cambiamo la scelta in 1
MACHINE **** Ok, hai scelto di prendere 1 fiammifero

[Sul tavolo ci sono 4 fiammiferi]
MACHINE **** Oh Human, la mia mossa e` prendere 3 fiammiferi

[Ora sul tavolo c'e` un solo fiammifero (eh eh ...)]
MACHINE **** Oh Human, la tua mossa? (rimuovi da 1 a 3 fiammiferi) ... 5
... no, scelta illegale: tsk, cambiamo la scelta in 1
MACHINE **** Ok, hai scelto di prendere 1 fiammifero

MACHINE **** Bip ... Bip ... Oh Human, hai preso l'ultimo fiammifero ... Bip ...
MACHINE **** Bip ... Bip ... Ho vinto :) ... Bip ...
Grazie della partita :)
```

(Poi guardate `fiammiferi3.c`, del quale l'output di una esecuzione era quello qui sopra).

**Disquisizione**
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. (Ma questo programma non lo abbiamo fatto).

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 realtà 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"*.

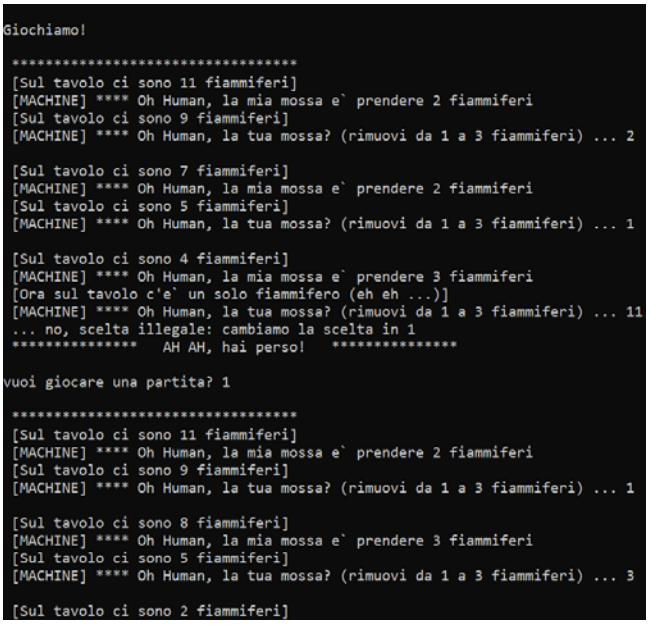
**Fine della disquisizione**

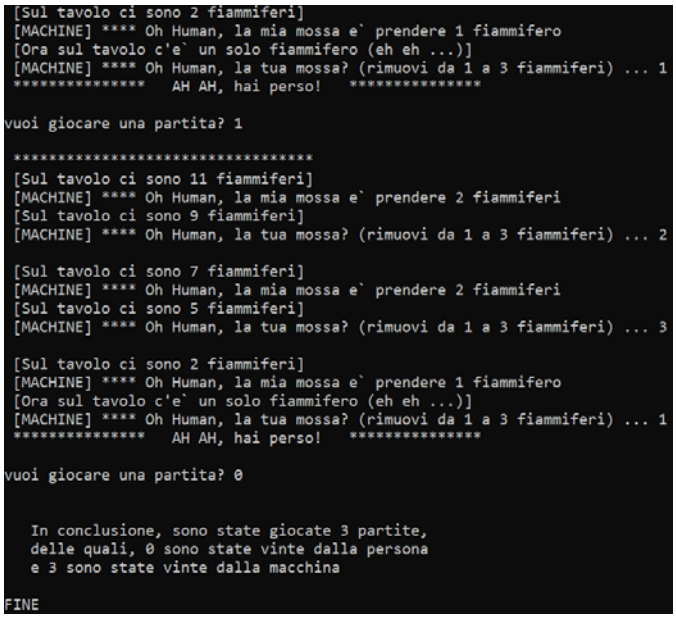
### *4.29. Rilassiamoci per migliorare ancora di più*

Dopo averci un po' pensato, rilassatevi modificando `fiammiferi3.c` (anzi … `mioFiammiferi3.c`, se avete chiamato così il vostro programma).

Le modifiche devono permettere di svolgere più partite nella stessa esecuzione del programma, tenendo il conto di quante ne ha vinte il computer e quante la persona.

Un esempio di output segue, ma dopo ci sono specifiche da leggere, prima di mettersi a programmare.




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() più pulita ...
- e una per giocare una partita (in pratica il programma precedente).

Prima scrivete l'algoritmo:

le variabili relative alla partita saranno nella funzione che la gioca, chiamata `unaPartita()`;

la `main()` avrà bisogno di variabili nuove

- 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 …

Provare a scrivere l'algoritmo del programma principale, in cui al momento giusto vengono chiamate le funzioni sopra accennate

segue un suggerimento per la main …
(ricordiamo che è lì che iniziamo a progettare le funzioni …)

**Suggerimento per l'algoritmo della main**

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() viene chiamata quando l'utente ha espresso la scelta di giocarne un'altra (la prima e` obbligatoria …)
   Che parametri deve ricevere? Pensarci e poi vedere dopo

   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`** è diversa da zero
   a. **`score = unaPartita(…);`** (pensare a come completare questa chiamata … con i necessari parametri attuali: da che dipende lo svolgimento di una partita? Dal fatto che ci sono due giocatori? Dal numero di fiammiferi inziali? Dalla dimensione della presa legittima? Dall'età dei giocatori? Dal numero di partite già giocate?
      Inoltre, cosa restituisce questa funzione? Restituisce un valore che indichi se ha vinto la persona o la macchina … )
   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 (tante vinte da chi …)

4) fine

Iniziare con la funzione main(), e le costanti, e l'include … nel file **`mioFiammiferi4.c`**

Inserite anche la chiamata a **`unaPartita()`** come riuscite a farlo. Ragionate sulla forma di questa chiamata. Quali sono i parametri attuali?
In cima, inserite il prototipo, secondo le riflessioni che avete fatto.

Una soluzione è in `fiammiferi4.c`

Poi qui sotto c'è un suggerimento sulla chiamata della funzione.

**Suggerimento sulla chiamata di unaPartita()**

In sunto
*"11 fiammiferi con prese da 3"* e *"15 fiammiferi con prese da 4"*,
sono giochi in cui la strategia di gioco è diversa.

… **`unaPartita()`** potrebbe avere come parametri **attuali (nella chiamata)**
e quindi **formali nella definizione della funzione**)
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.

Fatto il riassunto, analizziamo in dettaglio il punto 2-a) visto prima

**`score = unaPartita(…);`**
… da che dipende lo svolgimento di una partita?

- Dal fatto che ci sono due giocatori?
  No … il gioco e` per due giocatori; questo non e` un parametro per specificare comportamenti speciali della funzione … non serve farglielo sapere … lo sa per definizione

- Dal numero di fiammiferi iniziali?
  In effetti, se avete visto i riferimenti web … possiamo avere versioni diverse del gioco. Il gioco stesso e` parametrico e puo` essere giocato con diversi numeri di fiammiferi … e funziona in base a questo numero prestabilito: quindi si`, questo è un parametro caratterizzante il comportamento della funzione.
  E` vero che potremmo fare riferimento alla costante definita all'inizio del file, e non usare questo parametro. Ma è verosimile che vorremmo usare questa funzione anche in altri programmi, ed allora è bene progettarla in modo che non dipenda da elementi definiti in un particolare programma.
  (Gli altri programmi che volessero usare **`unaPartita()`** dovrebbero avere anche loro certe costanti, con i medesimi nomi di quelle in questo programma, e questa è una "dipendenza" che è meglio non introdurre …)
  Quindi sì … questo è un parametro attuale (**`nFiamm`**)

  Dalla dimensione della presa legittima? Stesse considerazioni appena fatte per **`nFiamm`**. Anche **`dimPresa`** e` un parametro attuale in questa chiamata.

- Dall'età dei giocatori? … andiamo …

- Dal numero di partite già giocate? La terza partita funziona diversamente dalla 12ma? (no)

  Inoltre, cosa restituisce questa funzione? Restituisce un valore che indichi se ha vinto la persona o la macchina … cioè **`0`** o **`1`** … in effetti è sempre zero, ma chissà`, un giorno le

cose cambieranno, quindi è bene che facciamo restituire questo risultato dalla funzione, anche se con un **`return 0`** …

Segue un suggerimento definitivo sulla chiamata

`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.

segue un suggerimento sull'algoritmo per **`unaPartita()`**

ma prima un’osservazione: avete provato il programma con diversi valori di numero fiammiferi e dimensione massima presa?

[
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**

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.c`**)

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, 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.

Aggiungere nel file che state usando (mioFiammiferi4.c, speriamo) i prototipi per

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

E poi scrivere queste funzioni …

Suggerimenti sulle intestazioni delle funzioni seguono

**Suggerimento**

```
int letturaMossaPersona(int, int, int);
```

La mossa legittima deve corrispondere ad una presa compresa tra il minimo e massimo valore consentito, AND non superiore al numero di fiammiferi rimasti sul tavolo.

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 perché 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

Ora potete completare il programma.
E` sensato riusare tutto il codice su cui avete lavorato per i precedenti programmi …

Quando il programma è fatto e testato, proseguite, se ve la sentite …

### *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", con 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.

Nel caso degli 11 fiammiferi, qualunque mossa M faccia l'avversario, se la successiva mossa (nostra, o della macchina) è 4-M, ricacciamo l'avversario in una configurazione perdente.

Lo stessoo vale per 15 fiammiferi iniziali, o 22, o 23, o 24 … provvisto che D sia 3
(21 no … è perdente …).

OK, abbiamo capito che D ha un ruolo importante, nel definire le configurazioni perdenti, e la mossa giusta.

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 sarà 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 più 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`** è più 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`** …)

Segue un esempio di output, ottenuto dal programma soluzione

```
Abbiamo 29 fiammiferi sul tavolo; ogni presa va da 1 a 5 fiammiferi.
Giochiamo!

----------------------------------------
 Cara unita` a carbonio, comincio io ...

[MACHINE] **** Oh Human, la mia mossa e` prendere 4 fiammiferi

[Sul tavolo ci sono 25 fiammiferi]
[MACHINE] **** Oh Human, la tua mossa? (rimuovi da 1 a 5 fiammiferi) ... 4

[Sul tavolo ci sono 21 fiammiferi]
[MACHINE] **** Oh Human, la mia mossa e` prendere 2 fiammiferi

[Sul tavolo ci sono 19 fiammiferi]
[MACHINE] **** Oh Human, la tua mossa? (rimuovi da 1 a 5 fiammiferi) ... 2

[Sul tavolo ci sono 17 fiammiferi]
[MACHINE] **** Oh Human, la mia mossa e` prendere 4 fiammiferi

[Sul tavolo ci sono 13 fiammiferi]
[MACHINE] **** Oh Human, la tua mossa? (rimuovi da 1 a 5 fiammiferi) ... 3

[Sul tavolo ci sono 10 fiammiferi]
[MACHINE] **** Oh Human, la mia mossa e` prendere 3 fiammiferi
```

```
[Sul tavolo ci sono 7 fiammiferi]
[MACHINE] **** Oh Human, la tua mossa? (rimuovi da 1 a 5 fiammiferi) ... 1

[Sul tavolo ci sono 6 fiammiferi]
[MACHINE] **** Oh Human, la mia mossa e` prendere 5 fiammiferi

[Ora sul tavolo c'e` un solo fiammifero (eh eh ...)]
[MACHINE] **** Oh Human, la tua mossa? (rimuovi da 1 a 5 fiammiferi) ... 1

[Ora sul tavolo c'e` un solo fiammifero (eh eh ...)]
**********   AH AH, hai perso!   **********

vuoi giocare una partita? (0/1))1

----------------------------------------
 Cara unita` a carbonio, comincio io ...

[MACHINE] **** Oh Human, la mia mossa e` prendere 4 fiammiferi

[Sul tavolo ci sono 25 fiammiferi]
[MACHINE] **** Oh Human, la tua mossa? (rimuovi da 1 a 5 fiammiferi) ... 5

[Sul tavolo ci sono 20 fiammiferi]
[MACHINE] **** Oh Human, la mia mossa e` prendere 1 fiammifero

[Sul tavolo ci sono 19 fiammiferi]
[MACHINE] **** Oh Human, la tua mossa? (rimuovi da 1 a 5 fiammiferi) ...
```

Suggerimenti sulle funzioni seguono

**Suggerimento sull'algoritmo di `unaPartitaQualsiasi()`**

è molto simile a quello visto per **`unaPartita()`**

Qui però, 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 è la **`mossaM`**, in qualsiasi turno della macchina, anche il primo..

Ci sarà 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 è sempre 1 …

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

segue un suggerimento sulla funzione **`maxMinoreDi()`**
cioè la funzione che deve calcolare il valore che è la prossima configurazione perdente da servire all'avversario …

**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`** è troppo grande (cioè ha raggiunto il limite dell'array … non ci sono più elementi dell'array da analizzare

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

quindi questa funzione deve ricevere sia l'array che il numero attuale di fiammiferi sul tavolo,

la sua chiamata:
**`maxMinoreDi(confPerdenti, actualFiamm);`**

Quando effettuiamo questa chiamata la macchina sta determinando la sua prossima mossa:

- siamo nella funzione **`unaPartitaQualsiasi()`**
- il valore calcolato dalla chiamata è la prossima configurazione perdente da servire all'avversario …

- Quindi effettivamente la chiamata andrebbe fatta per assegnare ad una variabile questa prossima configurazione (abbiamo chiamato **`proxConf`** la variabile che contiene questo valore):

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

Cosa si fa effettivamente con **`proxComf`** 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()`**

riceve

- il numero iniziale di fiammiferi stabilito per il tipo di gioco usato dalla **`main()`**
- la dimensione della presa massima
  - (i due valori qui sopra sono le caratteristiche del tipo di gioco dei fiammiferi che si gioca nel programma, e sono stabilite, nella **`main()`** 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
```

E` dura, lo so …
Ma prova a realizzare il programma completo in un file **`mioFiammiferi5.c`**

**`fiammiferi5.c`** contiene una implementazione del programma qui descritto.
Confronta, o cerca ispirazione, lì dentro.

Oppure cerca lì ispirazione …

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