Informatica I

6 – Progetti

9 Maggio 2013

Corso di Laurea in Matematica e applicazioni

Università di Camerino

A.A. 2012/2013

Progetto 0 – DB presenze

1 studente

Creare un oggetto `DBPresenze` che rappresenta un database per memorizzare presenze di individui ad eventi. Vanno anche creati gli oggetti `Individuo` e `Evento`.

Individuo(nome, cognome, stato) (Costruttore)

Crea un oggetto `Individuo` con una proprietà nome, cognome e stato (votante, non votante, …)

Evento(data, luogo) (Costruttore)

Crea un oggetto `Evento` con proprietà data e luogo.

Progetto 0 – DB presenze

DBPresenze(eventi,individui) (Costruttore)

Crea un oggetto `DBPresenze` che prende in ingresso un array di eventi e un array di individui.

setPresenza(idInviduo,idEvento) (metodo di DBPresenze)

Setta la presenza dell'individuo con `idInviduo` all'evento con `idEvento`. Gli identificativi corrispondo agli indici dell'array in cui individui ed eventi sono memorizzati. **Suggerimento:** servirsi di una matrice con num righe=num individui e num colonne=num eventi.

addEvento(evento) (metodo di DBPresenze)

Aggiunge `evento` all'array di eventi (e quindi aggiunge anche una colonna nella matrice delle presenze).

Progetto 0 – DB presenze

addIndividuo(individuo) (metodo di DBPresenze)

Aggiunge `individuo` all'array di individui (e quindi aggiunge anche una riga nella matrice delle presenze).

getIndividui() (metodo di DBPresenze)

Restituisce l'array degli individui

getEventi() (metodo di DBPresenze)

Restituisce l'array degli eventi

getPresenze() (metodo di DBPresenze)

Restituisce la matrice delle presenze

Progetto 0 – DB presenze

indexOfIndividuo(nome, cognome) (metodo di DBPresenze)

Restituisce l'indice dell'array in cui è memorizzato l'individuo con il dato `nome` e `cognome`.

indexOfEvento(data) (metodo di DBPresenze)

Restituisce l'indice dell'array in cui è memorizzato l'evento svolto nella `data` passata in argomento.

Progetto 1 – Convertitore temperatura

1 studente

Creare una funzione per convertire valori di temperatura in diverse unità di misura.

convertTemp(t, mode)

Restituisce la conversione identificata da `mode` della temperatura `t`. Mode può essere:

- "KC": kelvin to celsius
- "CK": celsius to kelvin
- "FK": fahrenheit to kelvin
- "KF": kelvin to fahrenheit
- "FC": fahrenheit to celsius
- "CF": celsius to fahrenheit

Progetto 2 – MCD

1 studente

Definire una funzione MCD per il calcolo del massimo comun divisore tra due naturali, secondo l'algoritmo di Euclide (mostrato nelle slides "Introduzione agli algoritmi").

MCD(a,b)

Restituisce il massimo comun divisore di a e b. Controlla anche inizialmente che a e b sono naturali (quindi positivi e interi).

Progetto 3 – Fibonacci

1 studente

Definire una funzione per il calcolo dei numeri di Fibonacci. L'n-esimo numero di Fibonacci è dato da

$$F_n = F_{n-1} + F_{n-2}$$

con

$$F_0 = 0 \text{ e } F_1 = 1$$

fibonacci(n)

Restituisce l'n-esimo numero di Fibonacci.

Progetto 4 – Brute primes

4 studenti

Realizzare una funzione che stampa tutti i numeri primi minori di un numero dato.

isPrime(n)

Restituisce `true` se `n` è primo. `False` altrimenti. La verifica di primalità è quella "brutale", ovvero bisogna verificare che `n` è divisibile solo per 1 e per se stesso (quindi che non esiste nessun divisore tra 2 e n-1).

Serve verificare fino a n-1?

brutePrimes(n)

Stampa tutti i numeri primi minori di `n`, servendosi della funzione `isPrime`. Per la stampa utilizzare la funzione `appendOutput(valoreDaStampare)` definita in utils.js

Progetto 5 – Macchina caffè

4 studenti

Creare un oggetto `CoffeeMachine` che simula un distributore che può erogare caffè o tè.

CoffeeMachine(prezzoCaffè, prezzoTè) (Costruttore)

Crea un oggetto **`CoffeeMachine`** inizializzando i prezzi (in centesimi) delle bevande. Naturalmente il credito corrente è zero.

insertCoin(value)

Inserisce una moneta e aggiorna il credito corrente. `value` rappresenta l'importo in centesimi e quindi può essere solamente: 5, 10, 20, 50, 100 o 200. Importi diversi da questi vanno ignorati.

getCredit()

Restituisce il credito corrente.

Progetto 5 – Macchina caffè

4 studenti

choose(beverage)

Sceglie la bevanda specificata nella stringa `beverage`. Se il credito è sufficiente, viene aggiornato e la funzione restituisce la `beverage` scelta. Altrimenti restituisce un messaggio "`Credito insufficiente`".

getChange()

La macchina rende il resto. Ovvero restituisce l'importo rimanente e lo azzera.

Progetto 6 – Cinematica

4 studenti

Creare un insieme di funzioni per calcolare il moto di un corpo.

uniforme(x0,v,times)

`times` è un array di istanti di tempo. Restituisce un array (della stessa lunghezza di `times`) in cui ogni elemento contiene la posizione del corpo all'istante indicato da `times`, secondo un moto rett. uniforme con velocità `v` e pos. iniziale `x0`.

uniformeAcc(x0,v0,a,times)

Similmente a `uniforme`, calcola il moto uniformemente accelerato con velocità iniziale `v0`, pos. iniziale `x0` e accelerazione `a`. In questo caso, restituisce un array bidimensionale che contiene sia posizione che velocità.

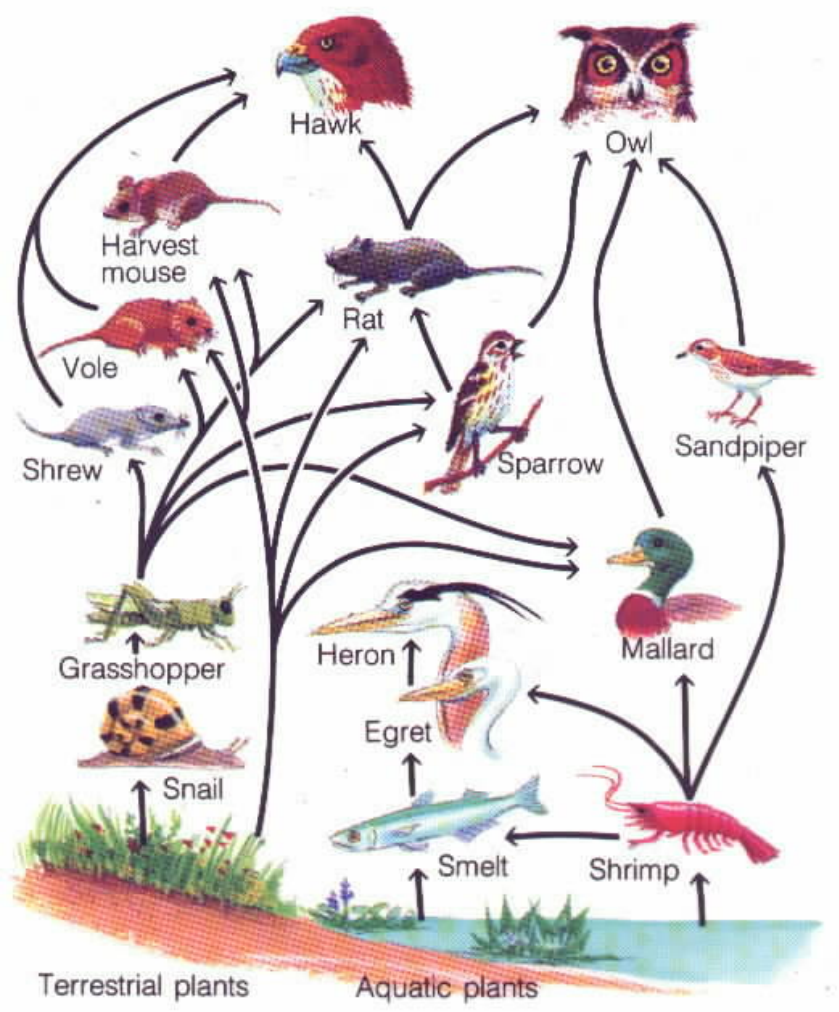
Progetto 7 – Food web

5 studenti

Creare un oggetto `Foodweb` che rappresenta una "rete alimentare" tra specie. Una food web può essere vista come un **grafo diretto** in cui un arco (i,j) indica che la specie i (preda) è mangiata dalla specie j (predatore).
In particolare va implementata una food web **quantitativa,** ovvero un grafo **diretto e pesato**, in cui il peso di un arco (i,j) rivela la quantità di i che è mangiata da j (**flusso**).

Grafo pesato

Si implementa come un grafo semplice (matrice adiacenza), con la differenza che, anziché 0 (assenza di arco) o 1 (presenza di arco), si memorizzano i pesi (reali non negativi) degli archi. Per convenzione, le prede sono collocate come righe e i predatori come colonne della matrice.



FoodWeb(nSpecies,maxFlow) (Costruttore)

Crea un oggetto `Foodweb` con un numero di specie (nodi) pari a `nSpecies`. I flussi (pesi degli archi) vanno inizializzati a un numero random tra 0 e `maxFlow` (anche se non realistico).

getWeb()

Restituisce la food web (ovvero la matrice di adiacenza)

getTotalInflow(i)

Restituisce la somma dei flussi in ingresso alla specie i.

getTotalOutflow(i)

Restituisce la somma dei flussi in uscita dalla specie i.

totalSystemThroughFlow()

Restituisce la somma di tutti i flussi della food web

getDietMatrix()

Restituisce la matrice dei flussi "ponderati", ovvero ogni flusso i->j è diviso per la somma dei flussi in ingresso al predatore j. In questo modo si ottiene quanto una preda i contribuisce alla dieta del predatore j.

Progetto 8 – Statistics

1 studente

Creare un oggetto `Statistics` che implementa semplici funzioni statistiche basate su osservazioni di una variabile random.

Statistics(nObs, min, max) (Costruttore)

Crea un oggetto `Statistics` che rappresenta una variabile random con un numero `nObs` di osservazioni. Le osservazioni vanno inizializzate a un numero random tra `min` e `max`.

getObs()

Restituisce le osservazioni

getMean()

Restituisce la media delle osservazioni.

Progetto 8 – Statistics

getVar()

Restituisce la varianza.

getSD()

Restituisce la deviazione standard.

Progetto 9 – Lotka-Volterra

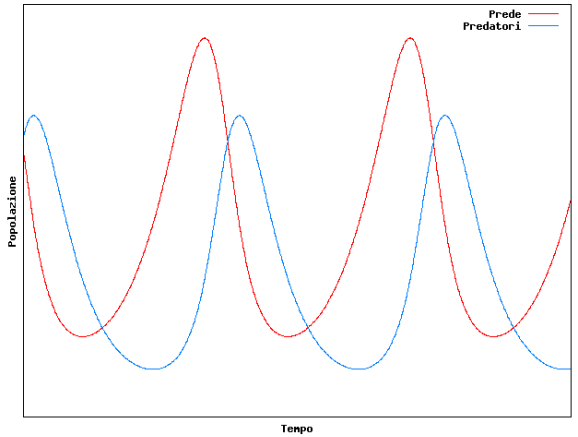
5 studenti

Creare una funzione per calcolare la popolazione di due specie in competizione tra loro, secondo il modello Lotka-Volterra (tempo discreto).

*N1[t+1] = N1[t] + a*N1[t] – b*N1[t]*N2[t] (prey)*

*N2[t+1] = N2[t] + d*N1[t]*N2[t] – g*N2[t] (predator)*

- Ni[t] rappresenta la dimensione della popolazione i a tempo t
- b, d rappresentano parametri di interazione tra specie
- a è il tasso intrinsico di crescita della preda
- g è il tasso intrinsico di decadimento del predatore
- a,b,d,g,N1[t],N2[t] >= 0.
- In assenza di prede, i predatori decadono esponenzialmente
- In assenza di predatori, le prede crescono esponenzialmente



Progetto 9 – Lotka-Volterra

LV(n1_0, n2_0, a, b, d, g, tFin)

Calcola l'abbondanza di prede e predatori secondo il modello Lotka Volterra.

- `n1_0` e `n2_0` sono i valori a tempo 0.
- `a`, `b`, `d`, `g` i parametri del modello.
- `tFin` il tempo finale di simulazione.

In particolare restituisce un array bidimensionale con una riga per specie, e una colonna per ogni istante di tempo da 0 a `tFin`.
Se i parametri in input non sono corretti (valori negativi) restituisce un messaggio d'errore.

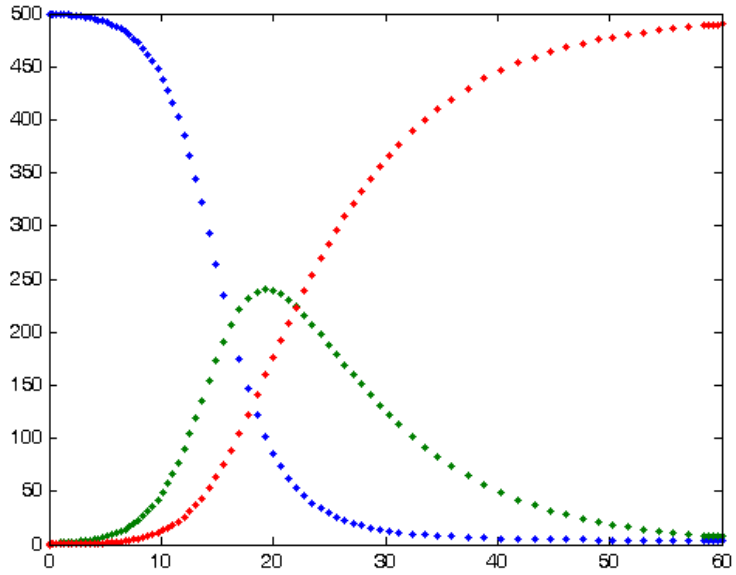
Progetto 10 – Epidemics

5 studenti

Creare una funzione per calcolare la progressione di un'epidemia secondo il modello SIR (Kermack, McKendrick), che distingue tra la porzione di popolazione suscettibile, quella infetta e quella recuperata (vaccinata).

*S[t+1] = S[t] –b*I[t]*S[t]* *(Susceptible)*
*I[t+1] = I[t] + b*I[t]*S[t] – v*I[t]* *(Infected)*
*R[t+1] = R[t] + v*I[t]* *(Recovered)*

- Quando un *susceptible* incontra un *infected*, diventa *infected*. Il tasso di incontro è dato dal parametro b.
- Un *infected* diventa *recovered* con un tasso *v*
- *S[t] + I[t] + R[t] = 1.* Si considerano quindi le percentuali sulla popolazione totale
- *b, v, S[t], I[t], R[t] >= 0*



Progetto 10 – Epidemics

SIR(s_0, i_0, r_0, b, v, tFin)

Calcola la progressione dell'epidemia secondo il modello SIR.

- `s_0`, `i_0` e `r_0` sono i valori delle variabili a tempo 0.
- `b`, `v` i parametri del modello.
- `tFin` il tempo finale di simulazione.

In particolare restituisce un array bidimensionale con tre righe (S, I, R), e una colonna per ogni istante di tempo da 0 a `tFin`.
Restituisce un messaggio d'errore quando l'input non è corretto, ovvero se:

- Uno tra *b, v, S[0], I[0], R[0] è < 0, oppure*
- *S[0] + I[0] + R[0] != 1*

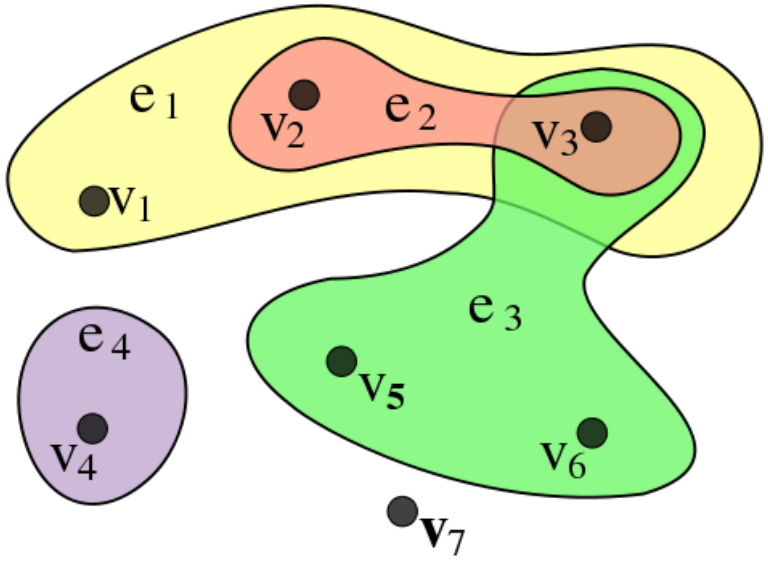
Progetto 11 - Hypergraph

5 studenti

Creare un oggetto `Hypergraph` che rappresenta un ipergrafo. Un ipergrafo è la generalizzazione di un grafo, in cui ogni arco può collegare un numero arbitrario di nodi.

In questo caso la matrice di adiacenza M ha una riga per ogni nodo e una colonna per ogni arco. $M_{n,e}$ è 1 se il nodo *n* appartiene all'arco *e*.

	e1	e2	e3	e4
v1	**1**	0	0	0
v2	**1**	**1**	0	0
v3	**1**	**1**	**1**	0
v4	0	0	0	**1**
v5	0	0	**1**	0
v6	0	0	**1**	0
v7	0	0	0	0



Hypergraph(nNodes, nEdges) (costruttore)

Crea un ipergrafo con un numero di nodi `nNodes` e un numero di iperarchi `nEdges`.

addToEdge(i,e)

Aggiunge il nodo `i` all'iperarco `e`.

removeFromEdge(i,e)

Rimuove il nodo `i` dall'iperarco `e`.

getGraph()

Restituisce l'ipergrafo, ovvero la matrice d'adiacenza

degree(i)

Restituisce il grado del nodo `i`, ovvero il numero di iperarchi che lo contiene

cardinality(e)

Restituisce la cardinalità (numero di nodi) dell'iperarco `e`.

isUniform()

Restituisce `true` se l'ipergrafo è uniforme. `false` altrimenti. Un ipergrafo è uniforme quando tutti gli archi hanno la stessa cardinalità.

rank()

Restituisce il rango dell'ipergrafo, ovvero la massima cardinalità di ogni iperarco.

dual()

Restituisce il duale dell'ipergrafo corrente, ovvero un oggetto Hypergraph i cui vertici e archi sono scambiati (si fa praticamente la trasposizione della matrice di adiacenza).

Gruppi

12 gruppi (e altrettanti progetti)**:** 5 gruppi da 1 (0-3 e 11); 3 gruppi da 4 (4-6); 5 gruppi da 5 (7-11). **Totale:** 42 studenti

Gruppo 0: Manzionna
Gruppo 1: Moretti
Gruppo 2: Menichelli
Gruppo 3: Antinori
Gruppo 4: Aquilanti, Giancamilli, Raponi, Saracchini
Gruppo 5: Marinucci, Alessandrini, Cotichini, Losa
Gruppo 6: Del Bianco, Moroni, Staffolani, Stefoni
Gruppo 7: Gianni, Perticarini, Romagnoli, Cetraro, Mogetta
Gruppo 8: Abruzzese, Caporalini, Molinelli, Mancini, Taffetani
Gruppo 9: Sbaffi, Luciani, Ballatori, Mandorino, Chiucconi
Gruppo 10: Montelpare, Lancellotti, Marzufero, Carucci, Abrami
Gruppo 11: Cicolini

Algoritmo scelta progetti

```
function assegnaProgetti(progetti, gruppi){
   while(gruppi.length>0 && progetti.length>0){
      var proj=Math.floor(Math.random()*progetti.length);
      var group=Math.floor(Math.random()*gruppi.length);
      print("Gruppo: "+(gruppi[group])+". Progetto: "+
(progetti[proj]));
      gruppi.splice(group,1);
      progetti.splice(proj,1);
   }
}
```

```
function assegnaProgetti5(){
       assegnaProgetti([7,8,9,10,11],[7,8,9,10,11]);
}
function assegnaProgetti4(){
       assegnaProgetti([4,5,6],[4,5,6]);
}
function assegnaProgetti1(){
       assegnaProgetti([1,2,3,8],[1,2,3,11]);
}
```

Ricapitolando...

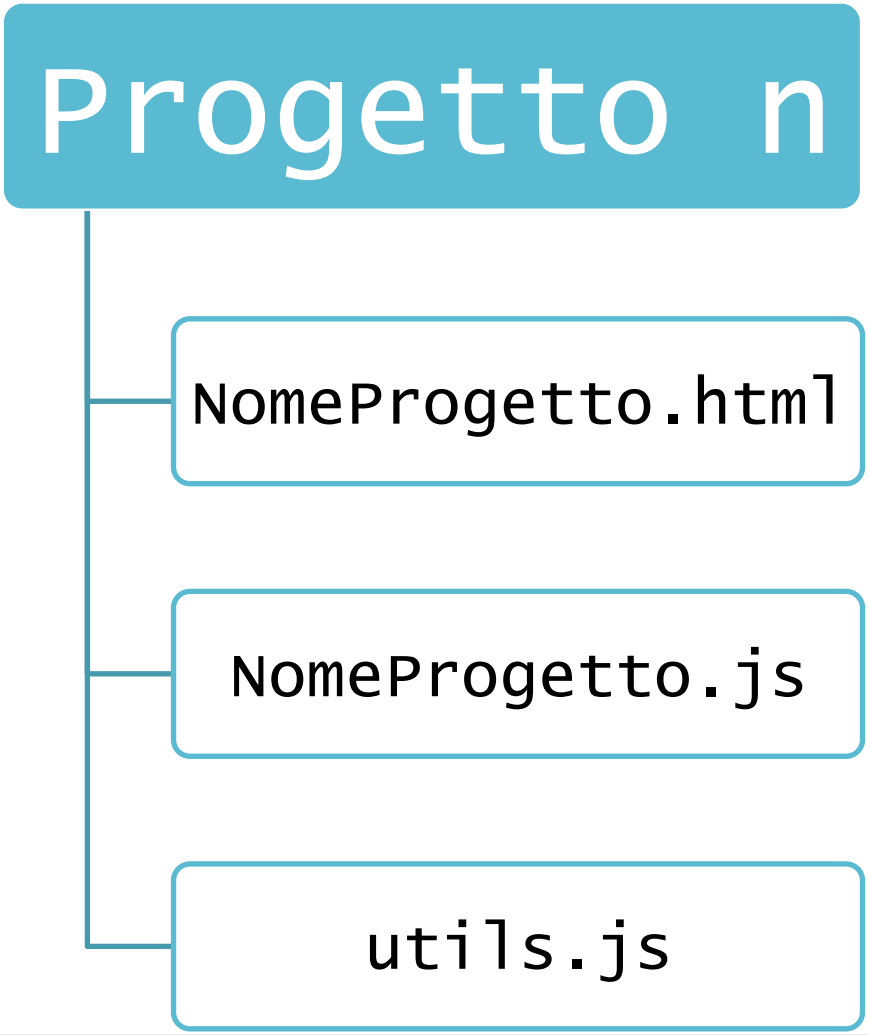
- Progetto 0: DB Presenze (Gruppo 0)
- Progetto 1: Convertitore temperatura (Gruppo 1)
- Progetto 2: MCD (Gruppo 3)
- Progetto 3: Fibonacci (Gruppo 2)
- Progetto 4: Brute primes (Gruppo 5)
- Progetto 5: Macchina caffè (Gruppo 6)
- Progetto 6: Cinematica (Gruppo 4)
- Progetto 7: Food web (Gruppo 8)
- Progetto 8: Statistics (Gruppo 11)
- Progetto 9: Lotka-Volterra (Gruppo 7)
- Progetto 10: Epidemics (Gruppo 9)
- Progetto 11: Hypergraph (Gruppo 10)

Strumenti di sviluppo

Avrete solo bisogno di un editor di testo per programmatori (che è diverso da un Word Processor, tipo Microsoft Word), e di un web browser.

Windows	MacOSX
EDITOR DI TESTO	
Notepad++ http://notepad-plus-plus.org/	**TextWrangler** http://www.barebones.com/products/textwrangler/
jEdit http://www.jedit.org	
Bluefish http://bfwiki.tellefsen.net/index.php/Installing_Bluefish	
WEB BROWSER (preferibilmente aggiornato)	
• **Mozilla Firefox (http://www.mozilla.org/en-US/firefox) + Firebug (https://getfirebug.com/downloads/)** •Google Chrome (https://www.google.com/chrome) •Opera (http://www.opera.com/)	

Struttura della cartella Progetto



Cartella progetto

Pagina HTML che fa da interfaccia con l'utente. Già pronta, carica il codice in `NomeProgetto.js`, e serve per testare i propri oggetti/funzioni.

File già contenente i template JS per i vostri oggetti, metodi, funzioni. Dovrete "riempirlo".

Alcune funzioni di utilità che servono nella pagina HTML. Guardare ma non toccare!

Relazione

Anzichè una relazione classicamente intesa, vi è richiesto di preparare alcune slides con le quali presenterete il lavoro svolto.

La struttura delle slides deve seguire i seguenti punti:

- **Prima pagina** (Nome progetto, componenti, A.A., Docente, Logo Unicam,...)
- **Specifica del progetto** (come presentata nelle mie slides)
- **Introduzione al problema** (background sulle nozioni necessarie per capire il problema da risolvere. Es. Progetto 7: introduzione alle food webs; Progetto 10: modelli epidemici)
- **Analisi del problema** (per ogni funzione/parte da implementare, spiegare e giustificare le scelte implementative, ovvero l'uso di una particolare struttura dati, o di un particolare comando di control flow, ...)
- **Implementazione** (copiate il codice JS con qualche ulteriore commento, se necessario)

Date presentazione progetti

- 20 Giugno ore 10
- 1 Luglio ore 10

Gia su ESSE3