Esercizi Programmazione
AA 2012-2013

Nicola Paoletti

6 Giugno 2013

1. Definire una funzione di comparazione `compareNumDec` per ordinare valori numerici in ordine **decrescente**. La funzione prende i parametri in ingresso `a` e `b` e restituisce un numerico > 0 se `a>b`; < 0 se `a<b`; $= 0$ se `a=b`. Da notare che se $\leq_{dec}$ è un ordinamento decrescente, vale che $a \leq_{dec} b \iff a \geq b$.

2. Definire una funzione di comparazione `compareArrays` per ordinare array in modo crescente rispetto al loro primo elemento. I parametri in ingresso sono due array a e b e restituisce un numerico > 0 se `a[0]>b[0]`; < 0 se `a[0]<b[0]`; $= 0$ se `a[0]=b[0]`.

3. Si vuole definire un oggetto `Complex`, che rappresenta un numero complesso. L'oggetto possiede due proprietà: `a`, parte reale; e `b`, parte immaginaria. Definire la sua funzione costruttore, che prende come in ingresso due parametri `a` e `b` con i quali inizializza le proprietà dell'oggetto. Nel caso in cui i parametri siano `undefined`, inizializza le due proprietà a 0.

4. Estendere l'oggetto `Array` con un metodo `empty` che "svuota" l'array, ovvero setta la lunghezza dell'array a 0. Ricordo che la sintassi per dichiarare un metodo `method` di un oggetto `Object` è:

```
Object.prototype.method = function(...){
  ...
}
```

5. Implementare una funzione `reverse` (che funziona esattamente come la funzione `reverse` dell'oggetto `Array`) che prende in ingresso un array `a`, e restituisce un nuovo array con gli stessi elementi di `a`, ma disposti in ordine invertito.

6. Creare una funzione `SSR`, che prende in ingresso due array numerici `a` e `b` e restituisce la somma dei quadrati residui tra i due array. Più precisamente restituisce

$$\sum_{i=0}^{a.length-1} (a[i] - b[i])^2$$

   Ricordo che l'elevamento a potenza a^b, in Javascript si ottiene chiamando il metodo `pow` dell'oggetto `Math`, ovvero `Math.pow(a,b)`.

Correzioni

1. è come la funzione per ordinare in modo crescente, solamente che in quel caso si restituisce `a-b`, qui `b-a`.

   ```
   function compareNumDec(a,b){
     return b-a;
   }
   ```

2. in questo caso, il modo in cui confrontiamo i primi elementi dell'array si deve servire dell'ordinamento naturale dei suoi elementi, in quanto non sappiamo a priori il tipo d'oggetti che contiene.

   ```
   function compareArrays(a,b){
     if(a[0]>b[0])
       return 1;
     else if(a[0]==b[0])
       return 0;
     else return -1;
   }
   ```

3. ```
   function Complex(a,b){
     if(a!=undefined)
       this.a=a;
     else
       this.a=0;
     if(b!=undefined)
       this.b=b;
     else
       this.b=0;
   }
   ```

   o in modo più compatto (vedere slides Javascript, esempio `Cerchio`):

   ```
   function Complex(a,b){
     this.a=a||0;
     this.b=b||0;
   }
   ```

4. basta settare la proprietà `length` dell'oggetto `Array` a 0. Naturalmente, tale metodo non ha bisogno di argomenti in ingresso, e non restituisce nulla.

   ```
   Array.prototype.empty = function(){
     this.length=0;
   }
   ```

5. ```
   function reverse(a){
     var r=new Array(a.length);
     for(var i=0; i<a.length; i++)
       r[a.length-1-i]=a[i];
   ```

```
  return r;
}
```

oppure si scorre al contrario

```
function reverse(a){
  var r=new Array(a.length);
  for(var i=a.length-1; i>=0; i--)
    r[i]=a[a.length-1-i];
  return r;
}
```

6. ```
   function SSR(a,b){
     var sum=0;
     for(var i=0; i<a.length; i++)
       sum+=Math.pow(a[i]-b[i], 2);
     return sum;
   }
   ```
   ```