Array e matrici

1

Scrivere una procedura ricorsiva che stampi i valori di un array in ordine inverso.

2

Scrivere una procedura che ritorni gli indici di riga e colonna del primo valore massimo in una matrice di interi generati randomicamente, dopo averla mostrata a video.

```
Matrice:
1 2 4 2
5 3 4 2
6 4 7 5

Valore massimo in [2, 2]
```

3

Scrivere una procedura che calcoli il prodotto tra due matrici quadrate inizializzate randomicamente con valori tra `[0, 10]` e stampi a video il risultato.

Ordinamento

4

Si vuole scrivere una procedura particolare per copiare il contenuto di un array di dimensione m (`arr_m`) in uno di dimensione n (`arr_n`), con $m < n$. Si nota subito che l'array di dimensione m non ha sufficienti elementi per poter riempire completamente l'array di dimensione n, perciò si decide di completare i valori mancanti generandone di nuovi:

- se `arr_m` è ordinato (sia crescente che decrescente), i nuovi valori saranno ottenuti "specchiando" gli m - *1* precedenti
- se `arr_m` non è ordinato, i nuovi valori saranno degli zeri

non si fanno assunzioni sulla contiguità dei valori nell'array `m`, inoltre i valori generati saranno sempre aggiunti *dopo* quelli contenuti in `arr_m`. Ad esempio:

```
int arr_m[5] = {1, 2, 4, 7, 8};
int arr_n[8];
```

l'array `arr_n` conterrà `[1, 2, 4, 7, 8, 7, 4, 2]`.

```
int arr_m[5] = {11, 7, 6, 4, 2};
int arr_n[14];
```

l'array `arr_n` conterrà `[11, 7, 6, 4, 2, 4, 6, 7, 11, 7, 6, 4, 2, 4]`.

```
int arr_m[3] = {1, 0, 2};
int arr_n[5];
```

l'array `arr_n` conterrà `[1, 0, 2, 0, 0]`.

5

Seguendo la procedura precedente, una volta ottenuti gli array con i valori generati, ordinarli utilizzando un algoritmo di ordinamento a scelta (SimpleSort, BubbleSort, MergeSort, QuickSort, ...).
Alternativamente, provare ad implementare un algoritmo di ordinamento definito dalle seguenti operazioni:

1. dato un array `arr[n]`, trova il valore massimo `max(arr) = m` contenuto al suo interno
2. crea un array `appoggio[m]` e lo inizializza tutto a zero
3. per ogni elemento in `arr`, incrementa di 1 la cella in posizione corrispondente al valore `arr[i]` in `appoggio`
4. per ogni elemento in `appoggio`, se il valore contenuto è maggiore di zero, stampa a video la posizione in cui si trova tante volte quante il valore che contiene

Ad esempio, dato

```
int arr[4] = {2, 1, 7, 2};
```

si dovrà creare

```
appoggio[7];
```

che, a seguito dell'operazione 3, conterrà `[1, 2, 0, 0, 0, 0, 1]`.
L'operazione 4, leggendo `appoggio`, stamperà

```
1 2 2 7
```

Attenzione agli indici!

Qual è la complessità di questo algoritmo? In quali casi può essere poco pratico o impossibile utilizzarlo?