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 presi in ingresso una matrice 3×4 e un vettore di dimensione 4, aggiunga il vettore ad ogni riga della matrice:

```
int Mat[3][4] = [
  [1,2,3,4],
  [5,6,7,8],
  [9,0,1,2]
];
int V[4] = [1,2,3,4];

int Res[3][4] = [
  [2,4,6,8],
  [6,8,10,11],
  [10,2,4,6]
];
```

4

Scrivere una procedura che calcoli il prodotto tra due matrici quadrate inizializzate randomicamente con valori tra `[0, 10]` e stampi a video il risultato. Si possono utilizzare più funzioni separate a supporto.

Ordinamento

5

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 $m < n$. I valori in `arr_m` devono essere inseriti dall'utente, mentre m e n vanno dichiarati come costanti.
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 precedenti
- se `arr_m` non è ordinato, i nuovi valori saranno degli zeri

non si fanno assunzioni sulla contiguità dei valori nell'array `arr_m`, inoltre i valori generati saranno sempre aggiunti *dopo* quelli contenuti in `arr_m`.
La procedura deve sfruttare l'aritmetica dei puntatori dove possibile e/o sensato.
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, 8, 7, 4]`.

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

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

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

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

6

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?

Extra

7

Scrivere una procedura che

- riempia di 1 e 0 una matrice di dimensione 10×10,
- la stampi a video
- dica quanti gruppi di 1 in celle vicine ci sono. Due celle sono vicine se sono sulla stessa colonna o se sono sulla stessa riga.

```
[
  1,1,1,1,1,0,1
  0,1,1,1,1,0,0
  0,0,0,0,0,0,0
  1,0,1,1,1,1,1
  0,1,0,0,0,0,0
  1,1,1,1,1,1,1
  0,0,0,0,0,0,0
]
```

In questo caso ci sono 5 gruppi di 1. In alto a sinistra c'è un gruppo di nove 1, in altro a destra c'è un gruppo di un uno, al centro a sinistra c'è un gruppo di un uno, in centro a destra c'è un gruppo di 5 uno, in basso c'è un gruppo di otto 1.

Bonus point per dire quali sono le dimensioni dei gruppi.

Modificare poi la procedura in modo che le dimensioni siano 100×100 e che la matrice venga stampata solo se la somma delle dimensioni sia minore o uguale a **20**.

Tips will be added later.

Nota

Per tutti i prossimi esercizi, cambiare i valori delle dimensioni degli array e delle matrici per provare diversi casi e verificare che il codice sia corretto.